Advances in Molecular Programming and Computing: Toward Chemistry as a New Information Technology

Report from a workshop in Copenhagen, May 2-4, 2013

1. Introduction

The computing revolution began over two thousand years ago with mechanical calculating machines such as the Antikythera mechanism. Subsequent developments produced sophisticated clockwork automata for controlling the machinery of the industrial revolution, and culminated in Babbage's remarkable design for a programmable mechanical computer. With the electronic revolution of the last century, the speed and complexity of computers increased dramatically. Using embedded computers we now program the behavior of a vast array of electro-mechanical devices, from cell phones to automobile engines to power grids. The history of computing has taught us two things: first, that the principles of computing can be embodied in a wide variety of physical substrates from gears and springs to transistors; and second that the mastery of a new physical substrate for computing has the potential to transform technology.

Another revolution is just beginning, one that will result in new types of programmable systems based on molecules. Like the previous revolutions, this "molecular programming revolution" will take much of its theory from computer science, but will require reformulation of familiar concepts such as programming languages and compilers, data structures and algorithms, resources and complexity, concurrency and stochasticity, correctness and robustness. It will be interdisciplinary, drawing inspiration, problems, and solutions from all the molecular sciences—molecular biology, chemistry, materials science, thermodynamics—and giving back tools for the analysis and synthesis of complex molecular systems. With molecular programming, chemistry will become the new information technology of the 21st century.

We envision that molecular programming will provide new insights and solutions to grand challenges in biology, chemistry, materials science, and medicine. In manufacturing it will enable fabrication of complex products from the bottom-up (e.g., programmable materials and devices grown from molecules); in biological engineering it will bring deeper computer science principles to the design of programmable subsystems within living cells. The knowledge gained will clarify the relationship between computation and the physical world—how information can be stored and processed by molecules, the limits of what can be computed and fabricated, and how quickly computation or fabrication can be done for a given energetic cost. While molecular programming will leverage existing tools from

computer science, control theory, electrical engineering, and bioengineering, it will also require new approaches to analyzing complex programs, circuits, and networks of molecules.

The seeds of this revolution are already being pursued in a number of research fields, variously described as synthetic biology, DNA nanotechnology, DNA computing, molecular programming, molecular computing, and information-based chemistry. Currently, the core concepts of molecular information technology are not yet recognized as an essential research subject within the computer science and engineering fields. However, we believe that for the field of molecular programming to flower, the intellectual depth of computer science and engineering must be brought to bear, and a unifying scientific research community must be built that develops the principles and practice of molecular information technology, with strong connections to other fields in which it will be applied.

2. Organization of the workshop and of this report

The primary objective of the workshop was to articulate a vision for the advancement of the information science and technology aspects of molecular chemistry and biochemistry, and how chemical and biochemical substrates might emerge as a new information technology in the coming century. Thus, the workshop considered advances ranging from molecular devices to DNA nanotechnology to synthetic biology as well as related areas involving information processing and programmability within engineered (and natural) molecular, chemical, and biological systems. Molecular programming was explored from theoretical, scientific, technological, and application perspectives—taking an interdisciplinary and international view to assess the current state, potential, and challenges of this new research area.

The secondary objective was to bring this emerging frontier more prominently to the attention of funding agencies worldwide, and to provide an unbiased assessment of its potential and challenges that will inform future decision-making. To ensure a global scope, throughout the planning stages we sought involvement from (and funding from) the US National Science Foundation (NSF), the Danish Council for Strategic Research, the UK Biotechnology and Biological Science Research Council (BBSRC), the UK Engineering and Physical Science Research Council (EPSRC), the Japan Society for the Promotion of Science (JSPS), the Council of Scientific and Industrial Research (CSIR) in India, as well as agencies in China, Canada, Germany, France, and Israel. This report is for their benefit. In particular, we aim here to explain the need for vigorous research activity in this area, with an emphasis on how researchers (and funding agencies) with deep experience in information science and technology will be integral to developing this frontier.

The tertiary objective was to expand the network of scientists and engineers working in molecular programming and to bring together a diverse community of researchers who will, over time, build this new field. The workshop was a meeting of minds. Participants were chosen by an 18-member strong program committee, chaired by Erik Winfree, of leading researchers from 8 academic disciplines and 11 countries (see the appendix for membership). The 42 participants spanned chemistry, physics, bioengineering, computer science, biochemistry, mathematics, robotics, electrical engineering, and biology; hailed from America, Denmark, England, France, Israel, Switzerland, Japan, Canada, India, China, and Germany; were at career stages ranging from postdoctoral fellowships to assistant professorships to tenured professorships to positions in industry and national funding agencies—but were unified by an excitement about the frontiers of programmable molecular technology (see the appendix for a list of participants).

A local organizing committee chaired by Kurt Gothelf, with support from the Danish Council for Strategic Research, arranged the workshop venue for a first day of talks at the historical Carlsberg Academy in Copenhagen, followed by two additional days of talks and discussion at the Magleaas Conference Center—a converted farmhouse in the beautiful countryside outside Copenhagen (see the appendix for the program schedule). For the second half of the workshop, participants formed working groups to discuss critical topics for molecular programming and computing: (1) theory for molecular information systems, (2) modeling, analysis, and specification, (3) applications in biology, biotechnology, and medicine, (4) applications in chemistry, physics, and materials, and (5) the role of technology. The leaders of those discussion groups (Manoj Gopalkrishnan, Masami Hagiya, Udi Shapiro, Andrew Phillips, Marta Kwiatkowska, Luca Cardelli, Chris Thachuk, Yannick Rondelez, Jessica Walter, Elisenda Feliu, Anne Condon, Fritz Simmel, Kurt Gothelf, and Eric Klavins) wrote reports summarizing their conclusions. These have been compiled by Erik Winfree (with occasional minor edits of style, clarity, and content) and appear as the following sections of this report. Thus, the opinions expressed in each section are not consensus opinions—they are the views of the authors of that section alone—but they reflect diverse input and have withstood measured debate. A summary of conclusions is provided in section 9.

3. Relating computer science and molecular programming

In retrospect, Alan Turing's machine presented in 1936 resembles the later-discovered molecular machines of the cell—polymerases and the ribosome—much more than the later-constructed electronic computers. Hence it is only a historical coincidence that basic concepts of computer science—computation and algorithms (including deterministic vs. non-deterministic; stochastic; probabilistic; approximate; digital vs. analog; serial vs. parallel vs. reactive); concurrency, communication and synchronization; abstraction, modularity and hierarchical design; compositionality and semantics; specification vs. implementation; fault-

tolerance and robustness—were all developed in the context of electronic computers and not of molecular computing systems. We contend that they are equally applicable to both.

Therefore we believe that an appropriate depiction of the intellectual landscape that relates computer science (which for clarity should be called in this context computing science) to electronic computers on the one hand, and to molecular computing systems (including living systems), on the other hand, is as specified in Figure 1.

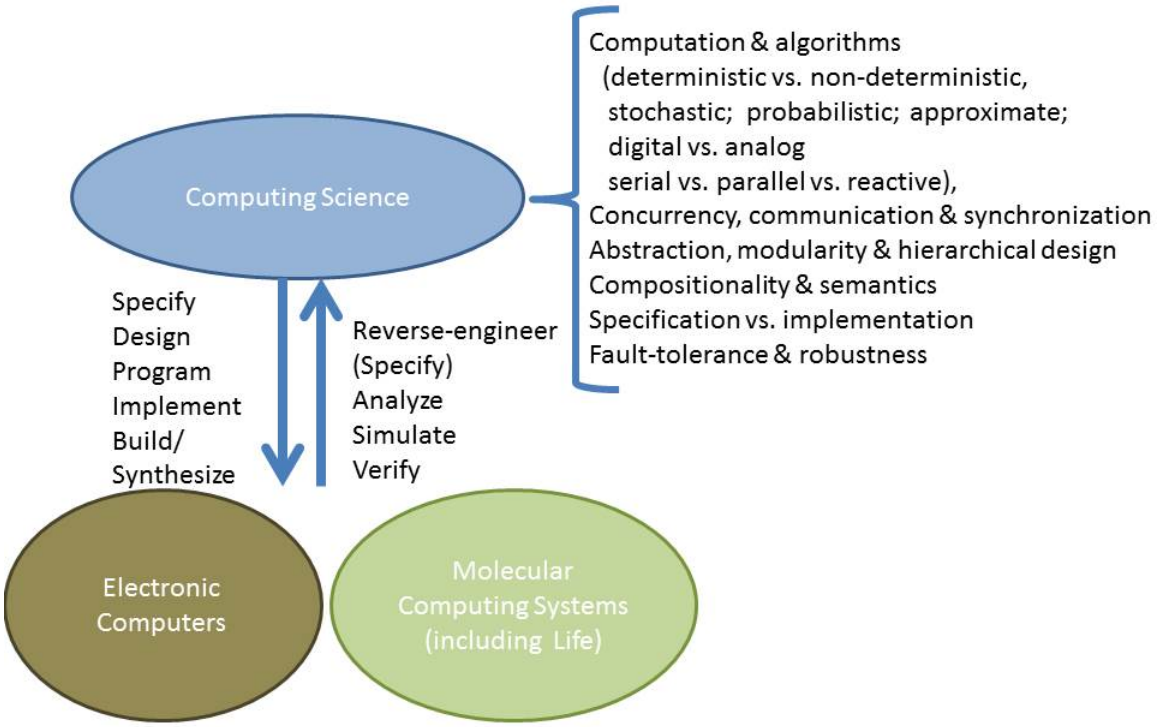

Figure 1: The relation between Computing Science, Electronic Computers and Molecular Computing Systems

Both the up and down arrows in Figure 1 apply to both substrates: electronic computers and molecular computing. Following the same logic, we view molecular programming and computing as the part of computing science applicable to the specification, design, programming, implementation, synthesis, reverse-engineering, analysis, simulation and verification of molecular computing systems (including Life), as depicted in Figure 2.

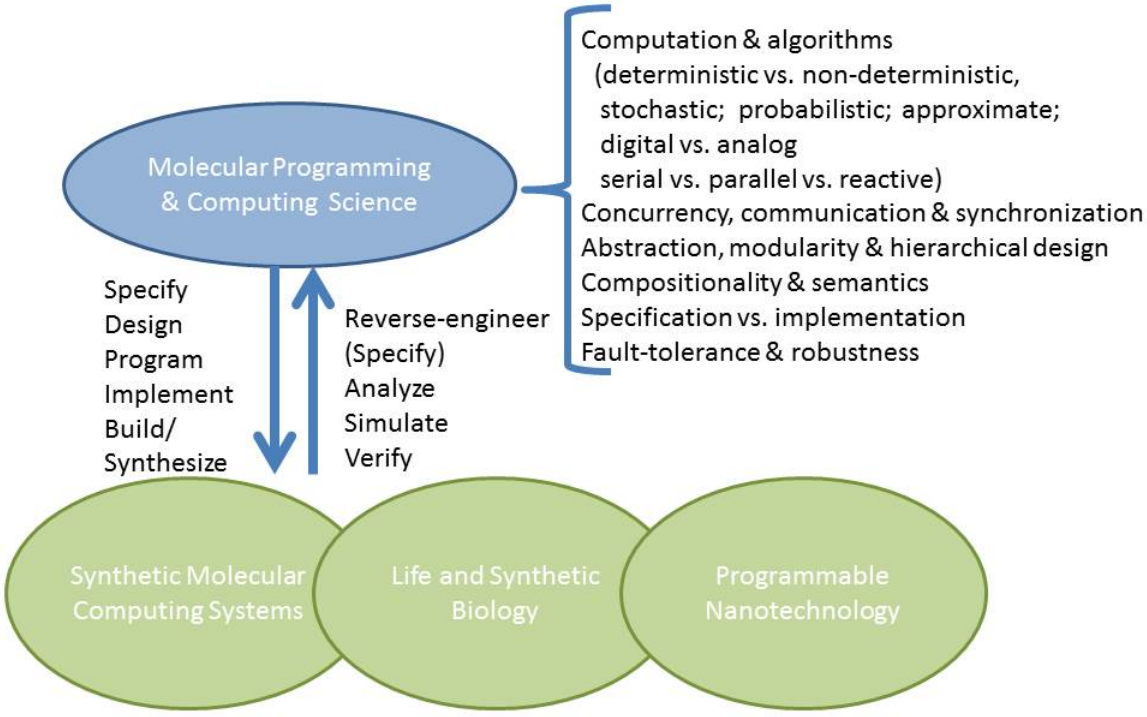

Figure 2: The relation between Molecular Computing & Programming to Molecular Computing Systems, Life and Programmable Nanotechnology

4. Theory for molecular information systems

Molecular programming and computing is an emerging inter-disciplinary field with connections to biology, systems and information technologies, chemistry, nanotechnology, physics and materials science. The grand vision of this field is to engineer molecular systems that approach living cells in sophistication. From the point of view of biologists, our endeavors are a part of systems biology, and specifically synthetic biology. The molecular engineer takes inspiration from biological phenomena, tries to reduce them to their essence, replicate them, and apply them as technology. When attempting to replicate sophisticated phenomena like molecular self-assembly and cascades of chemical reactions performing specific tasks, the community has been led to invent models to describe these phenomena at more abstract levels, making them amenable to theoretical analysis. Introducing such hierarchies of models has not only served to organize the daunting detail that goes into these implementations, but also organizes our understanding of what is possible and what is not possible within engineered molecular systems. Here we review the role of theory in molecular programming and computing, and explore future challenges.

Characteristics of models used in molecular programming and computing

We begin with a review of the types of models that have been considered in the molecular programming and computing community. There are currently a wide variety of models, each capturing different aspects of programmable molecular systems, at different levels of detail, and with different uses in mind. Thus, models for molecular computing can be classified according to three axes ``universal vs. problem-specific, ``coarse-grained vs. fine-grained'' and ``equilibrium/static vs. kinetic/dynamic'', which we explain before introducing the models themselves.

Universal vs. problem-specific. Some models can be classified as universal in the sense that they try to model some physical behaviors without any specific purpose. Thermodynamic and kinetic models of DNA hybridization are examples of such models—molecules with arbitrary sequences, be they biological or engineered, can be simulated. On the other hand, problem-specific models are often more limited but conceptually simpler, focusing on specific types of molecular phenomena or molecular mechanisms. For example, the tile assembly models (i.e., kTAM and aTAM) were introduced to analyze or predict the crystal-like process of DNA tile self-assembly, but cannot be used to study flexible or reconfigurable self-assembly processes. In some cases, a problem-specific model can be derived from a universal one by restricting it to a certain class of target systems or allowed mechanisms. Such models are "instantiations" of the universal model. For example, the kinetic model for the Visual DSD software can be derived from the universal kinetic model of DNA hybridization by focusing on strand displacement systems. It is usually the case that when restricting a universal model, some abstraction or approximation is also applied. It should also be recognized that "universal" and "problem-specific" are relative terms: a problem-specific model might be restricted to form an even more limited (but even more theoretically tractable) model, or a universal model might be seen to be a special case of a yet more general model. Thus, hierarchies of models exist. An important open question, in such cases, is how to rigorously establish connections between the models, so that theorems at one level can be applied at the other level.

Coarse-grained vs. fine-grained. Models representing different levels of physical detail can be related by abstraction or approximation. In general, a coarse-grained model is obtained from a fine-grained model by collapsing a set of states into a single state or multiple transitions into a single transition. Formally, if a behavior of the fine-grained model can always be simulated by a behavior of the coarse-grained model, the latter is called an abstract model of the former. Again, hierarchies of models occur naturally, and there are many open questions regarding how to relate such models rigorously. As an example, the aTAM is an abstract model of the kTAM. On the other hand, kTAM is justified (informally) by restricting the kinetic model of DNA hybridization to DNA tiles.

Equilibrium/static vs. kinetic/dynamic. Whether they are universal or problem-specific, coarse-grained or fine-grained, since models of molecular computation deal with dynamic physical phenomena, they can be classified with respect to the physical principles they employ. Some models are static in the sense that they only analyze or predict the final state of the target system, i.e., the equilibrium state. Other models are dynamic in the sense that they also analyze or predict dynamic behaviors of the system. Equilibrium models often can be analyzed using powerful theoretical tools and simplifying principles, but have limited behavioral repertoire; in contrast, kinetic models are often challenging to analyze but have more powerful capabilities.

Common models used in molecular programming and computing

Thermodynamic and kinetic models of DNA hybridization. Thermodynamic and kinetic models of DNA hybridization have been well studied, and algorithms and tools for sequence-level structure design have been developed (e.g., NUPACK). Those models were first introduced to predict the secondary structure of a single RNA molecule, but they have been adapted and refined to predict the secondary structure of multiple strands of DNA molecules. They have been improved in quality, speed and estimation, partly due to good estimation of thermodynamic and kinetic parameters. The kinetic model can even predict multi-strand folding pathways. As for strand displacement, some coarse-grained models have been derived from the general kinetic model.

There remain some challenges for these models. Firstly, obtaining better estimates of kinetic and thermodynamic parameters for DNA is still a challenge. It might be the case that our themodynamic models are too complex to allow better estimates and should be simplified. Secondly, kinetic simulations are slow in general. We need good coarse-grained models for DSD, but it is not trivial to test for accuracy of such coarse-grained models relative to a fine-grained model.

DNA strand displacement models. In molecular programming of DNA, the DSD (DNA strand displacement) model has been used for scalable design of logic circuits, artificial neural networks, dynamic DNA devices, walkers, etc. A variety of distinct formulations of the idea have been proposed. All are based on DNA hybridization with toehold-mediated strand displacement, which drives and controls transformation of DNA complexes. Reactions in the model are energy-efficient and can be even reversible. Since DNA complexes in the model are restricted, they have limited potential for unwanted strand interactions. Some higher-level (in this case more restricted) models, such as that of seesaw gates, address causes of error including signal degradation. For DSD, prototype compilers and tools for verification and visualization have been developed. For example, the Visual DSD simulator can predict reaction pathways in the model.

There also remain many challenges for DSD. An important one is to develop theoretical approaches to overcome barriers to scaling up experimental

implementations of DSD systems. Complexity barriers (e.g., the number of molecular species needed to implement a function) might be addressed by new ways of thinking about stochastic, parallel, always-interacting models of computation. Biophysical barriers (e.g., erroneous displacement pathways and spurious binding that slows progression of reactions) might be addressed with fault-tolerance and error-correcting system designs. The barrier of molecular sequence design (e.g., DNA sequences that with high probability fold, self-assemble, and react according to specification) might be addressed through a better understanding of the combinatorial structure of nucleic acid energy landscapes.

Chemical reaction networks. Chemical reaction networks (CRNs) can be described with various choices of dynamics: mass-action kinetics, hill kinetics and other pragmatic choices, the chemical master equation, other non-deterministic dynamics, etc. In practice, all of these dynamics choices are actually used, depending upon the application. The reaction network itself is an abstract graphical model. The field of "chemical reaction network theory" has established that a lot of information about the qualitative behavior of trajectories can be gleaned just from the graphical structure, even without knowing the rate constants, or even the precise dynamical model. Though CRNs are frequently used in the molecular programming community, the ideas of chemical reaction network theory have still not found application.

Soloveichik and others have argued that every chemical reaction network can be implemented by idealized DNA strand displacement reactions. Moreover CRNs have deep connections with the traditional theory of computation—Petri nets are effectively equivalent to CRNs, although the nature of questions asked often differs—and more recently Soloveichik and others have established connections of CRNs to certain models in distributed computing.

One such connection is given by defining a function $f(x) = y$ using a chemical reaction network, where x is the number of copies of the input species and y is the number of copies of the output species. It has been shown that this unary computation can define any semi-linear function, and the unary computation with error is even Turing complete. One can also define the binary computation by a chemical reaction network, whose properties have also been investigated well.

(It may be interesting to note that while DSD is an instantiation of CRN, DSD can also implement CRN. In this case, the implemented chemical reaction network can be considered as an abstract model of the implementing strand displacement system.)

Tile assembly models: kTAM and aTAM. Winfree's tile assembly models, kTAM and aTAM, are examples of problem-specific models in molecular computation. The question was: how can one form shapes and patterns using molecular self-assembly? Taking inspiration from Wang tilings, Winfree specialized that model to take into account the special needs of DNA-based tiles. The model has been shown to be universal for both computation and construction, and to have interesting fault-tolerance and self-healing capabilities.

Other models. Some models deal with membranes or compartments. Models proposed in the membrane computing community are examples. Process calculi, such as the π-calculus, also allow nested structures that can model systems with membranes. These models are mainly applied to analyze or predict cellular behaviors in the field of systems or synthetic biology, but will become important if compartmentalization is introduced to molecular computation as discussed in the next section.

Not all proposed models will match reality. It is very important to weed out the models that don't work from the ones that do work. The molecular programming community has to give some thought to empirical model validation. This is happening at the scale of some individual labs, but not yet at a community level.

Challenges for theory

The field of molecular programming and computing is still quite young. New experimental techniques are being invented rapidly, while there are other phenomena that look tantalizing to experimentalists, but for which we lack ways to think about them systematically. Hence, there is a need to invent many new models so that the precise language for thinking about such phenomena is extended. Here are some areas that to us look particularly in need of the development of new models.

DNA origami. DNA origami has been one of the great successes of the molecular programming approach. However, we lack a systematic way to think about DNA origami. It would be nice to have a theoretical model that predicts its yield, its time to assembly, its kinetics, etc. There have been some initial attempts at describing folding pathways. One crucial aspect is that cooperativity between various molecules appears to play a key role in the formation of DNA origami. A challenge is to come up with a theoretical model that can capture such cooperativity.

Development and differentiation. Biological systems are remarkable at creating shapes and patterns. The process of development from a single cell to an entire organism is one of the miracles of life. Differentiation of cells is a key step in this process of development. Cells that start off as identical, and formed by division of the same mother cell, start expressing different proteins in a cooperative manner, specializing to form different parts of the organism. This process is a possible source of inspiration for the molecular programmer. It seems intimidating to think about this process ab initio. What is sought is an abstract combinatorial model that distills out key aspects of this process, but is simple enough that it can be implemented, say with strand displacement cascades.

Geometry and topology. A cell is not a well-mixed environment. It has a lot of geometric structure, and perhaps it derives some of its power from this structure. On the other hand, many of our models abstract away all geometry and topology.

There is room for a model that can talk meaningfully about the geometry of reaction compartments, and how that can be used in a clever manner to achieve efficiencies. It might perhaps be easier to start with a theory that only keeps topological information, say by modeling the space as a graph.

Locality. One can rephrase geometry and topology as "locality" because the geometric structure in a cell brings reactants of a chemical reaction close to each other and makes the reaction more efficient. On the other hand, there should be mechanisms to selectively transport or diffuse molecules across geometry or topology. (This aspect is related to that of "walkers and molecular robots" below.)

Exquisite chemical detection. Exquisite chemical detection is the problem of cheap and rapid detection of the presence or absence of small numbers of molecules. This problem is simple enough to form a good test case for a complexity theory where one would attempt to prove an impossibility result that says that fixing the volume, given a certain amount of energy, and a certain amount of time, there is a limit to the smallest concentration that can be detected.

Walkers and molecular robots. There has been much interesting work on molecular walkers and molecular robots. However, there have been few theoretical models that help us understand the tradeoffs and limitations in such systems.

Leaks. In practice, when we put molecules in a reaction vessel, we witness many undesired interactions. There has been some preliminary attempt to model leaks with the notions of "saturated" and "catalytic" networks, but there is still room for a more mature theory of leaky reactions. Further, there is a meta-question of how robust different models are in the presence of such leaks.

Resources and performance metrics

We briefly list some resources that are of concern when performing experiments in molecular programming and computing.

Time. This is the biggest concern for many areas of molecular programming and computing. Reaction kinetics is much slower when compared to electronics, and is limited by diffusion, but that is not the right comparison. We should instead be comparing with the time scales of performing comparable tasks in biological systems. For example, DNA walkers are still much slower than kinesin walkers, and DNA amplifiers usually run on time scales of hours, as opposed to minutes in biological systems. In any case, the speeds of individual steps may be less important than the scaling properties of system designs as the number of components increases. The number of macromolecular components (individual proteins, RNA, and DNA molecules) in biological organisms such as mammals easily exceed Avogadro's number, 6×10^{23}, and thus a "small" difference in the scaling of the time needed for a molecular algorithm (say, from $O(n)$ to $O(\sqrt{n})$) could have a huge effect.

Volume. We mean the volume of the reactant vessel required to carry out the experiment. This is roughly analogous to space complexity in conventional computer science settings.

Circuit size and program size. By this we mean the cost of setting up the experiment. This includes the cost of preparing all the initial species, and other preparations required to set up the experiment. In particular, the number of different molecules required should scale gracefully.

Energy. This is an interesting metric to look at because of the amazing energy efficiencies obtained by biological systems. If we can come close to such efficiencies, then that could lead to many niche applications. There are potentially deep connections to the theory of reversible computing.

Robustness to noise and uncertainty. Both the assumptions of the model, and individual implementations within the model should be resilient to noise and uncertainty, whose nature is set by the application in mind.

Algorithms and complexity theory in molecular computing

Once a model has been agreed upon, the following theoretical questions emerge naturally. What are the tasks that can be expressed within this model? On the flip side, what tasks are impossible within this model? This relates to computability and uncomputability. We can ask this question in the presence of resource constraints, and that leads to resource-constrained algorithms and complexity theory for these models.

An important question, which has only begun to be explored, is how the field of theory in molecular computation is related to the broader field of information sciences and systems sciences, how deep results from those fields can be applied in the context of molecular programming and computing, and in particular how ideas from molecular programming and computing might contribute to these other areas. We expect that there are many natural connections to system science, circuit theory, computer architecture, complexity theory, information theory, coding theory, robotics, and the study of algorithms, to name a few promising areas.

5. Modeling, analysis, specification, and design

This research aims to devise programming languages, techniques and software tools to support the activity of molecular programming, with the high-level goal of achieving automatic compilation of a specification of a molecular system down to the set of components that can be implemented physically at the molecular level and executed. Typical functions of the resulting molecular programs are similar to digital/electronic systems that control physical processes, and include biosensing,

actuation, information processing and assembly, except that they are realized at the nanoscale.

The distinctive aspects of this enterprise, in comparison with natural sciences, is that, rather than striving to understand the physical reality, we wish to produce executable specifications of molecular systems that can directly interact with biological/chemical entities, for example those featured in medical applications, nanoscience and nanoengineering. This is analogous to the motivation for hardware description languages, e.g. VHDL, which are refined automatically, through a series of intermediate abstractions, down to a detailed physical implementation, e.g. an integrated circuit. The design and synthesis of electronic systems is supported by EDA (Electronic Design Automation) tools, which are now needed for molecular systems.

At its core, the research into programming languages for molecular programming constitutes the science that enables the construction of faithful but abstract models for molecular programs; that exploits a broad range of analysis techniques that allow us to ensure predictability and reliability, analyze performance and resource usage, and establish whether correctness requirements are satisfied; and that produces CAD tools to support model construction and analysis, including automated compilation directly into the physical systems. As was the case with electronic computers, successful development of molecular programming languages and EDA tools will (by definition) open up the use of this technology—from being accessible only to experts in the device- and machine-level intricacies, to researchers and engineers in other fields who have only a high-level understanding of the tasks they need the molecular system to perform. It will revolutionize our ability to engineer larger and more complex molecular systems.

At this time, it is possible to identify several key areas that are ripe for development and have the potential for large impact on the field:

- Design of programming languages that capture abstractions relevant to molecular devices, for example DNA strand displacement, molecular motors, localized hybridization, self-assembly, etc.
- Rigorous semantics for those programming languages, as supported by existing knowledge in theoretical computer science, which must be augmented by a variety of quantitative features (specifically, stochasticity, continuous/hybrid behaviors and their approximations, thermodynamics, kinetics, resource usage, e.g. energy) that have not been central to computer science.
- Compositional design methodologies and CAD tools, in order to support effective processes to engineer systems from independently specified components.
- Compiler technology, including design and implementation of intermediate language abstractions and algorithms for synthesizing molecular circuits.

- Analysis techniques and software tools for molecular programs, which include stochastic and deterministic simulation; methods for inferring models from experimental observations; state-based abstractions and exploration methods; behavioral theories and equivalences; quantitative approximation techniques; and automated and/or interactive verification methods against given correctness requirements.

Existing tools

The past few years have seen the development of a wave of new tools that are enabling molecular programming research, together with a variety of applicable tools from chemistry and systems biology.

- *Thermodynamics level:* analysis and design of DNA sequences at the secondary structure level, as implemented by e.g. NUPACK or RNAsoft.
- *Kinetics level*: simulations of secondary structure folding pathways, as implemented by e.g. Multistrand; simulation at the level of coarse-grained molecular dynamics, as implemented by e.g. oxDNA.
- *Structural level:* specification and analysis of three-dimensional structure of complex self-assembled nanostructures such as DNA origami, as implemented by e.g. caDNAno and CanDo.
- *Domain level:* stochastic simulation and analysis of DNA Strand Displacement Systems, as implemented by e.g. Visual DSD; including localized hybridization and reaction-diffusion systems; (probabilistic) model checking and debugging of DSD systems, supported by PRISM tools.
- *CRN level:* chemical reaction networks are supported by stochastic simulation tools, e.g. Gillespie and StochSim; deterministic simulation of mass-action kinetics, e.g. GEPASI and COPASI; and also probabilistic model checking against temporal logic properties, e.g. using PRISM.
- *Reactive-systems level*: a variety of process algebra and hybrid systems tools are applicable at this level, e.g. SPiM and kappa.
- *Multicellular systems level*: the behavior of cells and colonies of interacting cells can be specified and simulated using programming languages such as gro or CellModeller.

Goals for future developments

The following are major technical goals that concern modeling, analysis and specification:

- Though existing process-algebraic languages have proved useful for describing complex molecular systems, much more effort is needed to design high-level languages for emerging experimental phenomena and methodologies, including localization, 3D, geometry and mobility, together

with associated rigorous semantics, equivalence/refinement notions, and compositional behavioral theories.

- Despite progress made towards modeling well-mixed molecular systems, for example using stochastic pi-calculus, there is an urgent need to develop quantitative theories for the different levels of abstraction hierarchy, and from them devise algorithms, techniques and tools to support the design of predictable molecular systems.
- Stochastic simulation is known to be computationally intensive and their performance is poor for molecular systems that we wish to design and analyze today. We need much more efficient simulation techniques, for example those based on multi-scale simulation that have shown promise.
- State-based abstractions of molecular programs have the potential to enable analysis of correctness of the computation performed by the molecular program, for example deadlock, presence or absence of a given even, or termination. State-space exploration techniques can assist in establishing such correctness requirements, but state-space reduction techniques are necessary to speed up the analysis.
- Formal verification techniques, such as automated verification via model checking or interactive theorem proving, are able to establish, via a systematic exploration of the model or mathematical proof, that a given correctness verification is satisfied for a molecular program. Their limitation is the size of the state-space of the model that can be handled, and therefore scalable verification techniques are a major goal of this research.
- In addition to being able to perform verification of molecular programs against requirements, an important question is whether it is possible, given a specification, to automatically synthesize a program that is guaranteed to satisfy the specification. This approach would ensure correct-by-construction designs, and is in its infancy.
- Since the models of molecular programs typically include quantitative and possibly continuous aspects, for example stochasticity and energy, to facilitate their analysis one must apply abstractions and (numerical) approximations. This raises the question of the level of precision, including accuracy and error bounds, that the analysis method can inherently guarantee, in turn impacting the predictability of the molecular program's behavior.
- A major challenge is to integrate all abstraction levels (from thermodynamics to sequence to CRN and reactive systems), and to achieve fully automatic compilation from high-level specifications to physical structures, with analysis enabled at each level and connected across levels (by relying on compositionality of the designs and substitutivity of component specifications for their implementations to facilitate analysis).

The Holy Grail: A hierarchical and compositional model of whole organisms from the molecules and up

Ultimately, we believe that the science of molecular computing will reach a level of understanding, precision, and proficiency so that, combined with the best knowledge of molecular biology, it would be able to provide a hierarchical and compositional description of multicellular organisms—or purely synthetic molecular systems of comparable complexity—starting at the molecular level yet exhibiting correct behaviors at the organism level.

A key milestone to achieving this very long-term goal is embracing the key roles of specification vs. implementation in biology. Consider a cell of a particular type. A full description of the "implementation" of the cell at the molecular level would require describing all cell molecules and their interaction. A faithful description would entail predicting the correct behavior of the cell as observed from the outside: It would predict correctly how the cell would respond to sensing molecular signals, e.g. by releasing its own signals, dividing, differentiating, or performing apoptosis. However, if we fully understand the cell at the molecular level, we could, in principle, replace the detailed description of the cell's molecular implementation by a simpler specification of the cells behavior. Such a behavioral specification would in effect describe what the cell does, not in terms of its internal mechanisms, but in in terms of its behavior as visible from the outside. It may be fair to say that until we can come up with a behavioral specification of this cell type (or of any other biological system or organism, for that matter) we cannot claim we fully understand that cell (or system or organism).

Having correct compositional behavioral specifications of biological components, e.g. cells, is essential not only for understanding them but also for analysis and simulation of the whole organism. We cannot hope to ever simulate full organisms at the molecular level. But if we can build and verify hierarchical specifications of biological systems, where for each element in the hierarchy we have both an implementation in terms of its subcomponents lower in the hierarchy, as well as a compositional specification of the element that can be verified to have the same behavior as the implementation, then we can use that behavioral specification to simulate the behavior of the elements further up the hierarchy. See Figure X.

It is essential that specifications be compositional, so that it can be used to build subsequent specification up the hierarchy. It is important to note that even though components could be high level, their interaction might need to be described at the molecular level. As an extreme example consider modeling an ant community. While the specification of each ant can be composed from ant organs, the communication between ants, at the pheromone level, still must be described at the molecular level.

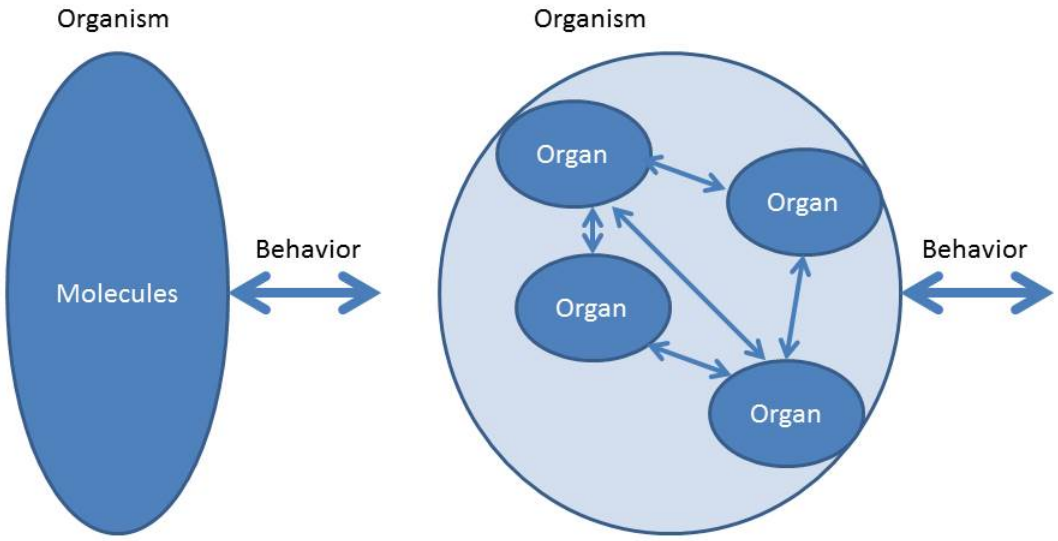

Figure X: Unstructured vs. Hierarchical and Compositional Specification

What opportunities for the field are within reach as the next research steps?

- For *molecular programming languages*: devise languages and their quantitative semantics for localized hybridization, integrating enzymes into DSD, geometry (based on distance between components), origami designs, molecular walkers, and microfluidic devices (for manipulating vesicles, based on brane calculi).
- For analysis via *stochastic simulation*: improve efficiency by integrating ODE techniques within stochastic simulation.
- For analysis via *automated verification*: for non-probabilistic models, utilize SMT (Satisfiability Modulo Theory) techniques for efficiency improvements; and for probabilistic models, improve scalability by devising abstraction-refinement schemes in conjunction with symbolic techniques.
- For *synthesis* from specifications, focus on the simpler problem of parameter synthesis, where the goal is to compute the ranges of parameters for which a given specification is guaranteed to hold; and parameter inference from experimental data, using Monte Carlo techniques.
- For *integration*, focus on common execution engines that support multiple languages and interchange formats.

The main scientific challenges are the scale, complexity and the range of quantitative aspects that have to be handled by the analysis methods. There is also a software challenge: how to relate and integrate across different levels of the abstraction hierarchy. The criteria for evaluation of success will include the rate of improvement in scale and complexity; for quantitative aspects, the accuracy of predictions; and the efficiency of software tools.

6. Applications in biology, biotechnology, and medicine

Molecular computing has the potential to transform many areas in life sciences, by virtue of being able to control living matter with unprecedented sophistication and precision. This capacity is by and large hypothetical, because living matter is very complex, evolving and noisy. Thus it is harder to control than man-made artifacts. Nevertheless, with sufficient investment of effort into the theoretical and practical aspect of molecular programming and computing, we expect with high degree of certainty that this vision will become reality.

Impact on basic biological science

We think that progress in molecular programming and computing will not only provide new technological tools for biological discoveries, but will generate new biological knowledge in its own right. Perhaps the most important anticipated contribution is in "bottom-up" construction of life-like processes that are decoupled from specific biochemical implementation we find in living objects. For example, concepts like information storage in biopolymers; transducing one biopolymer into another; complex interacting networks that support replication or response to environmental changes; and more can be analyzed and reconstructed from purely non-living molecular blocks. Moreover, information-processing networks that do not have natural analogs could be designed and built. Together, these advances will bring closer the concept of "synthetic life" and ultimately make us understand what are the key functional features of life.

Likewise, moving beyond recapitulating natural processes will teach us about the true possibilities of molecular information processing. For example, an interesting question is whether purely molecular networks can learn (as opposed to networks of cells that can learn). Another intriguing question is whether genetic material can be computed, rather than simply copied from the previous generation. In principle, updating the DNA sequence can be performed in a programmable fashion using Turing-like computation.

Last but not least, reconstructing the function of natural pathways in simplified cell-free or buffer environments can be a powerful tool to verify and generate hypotheses about these pathways.

Impact on applied biology

Diagnostics and biosensing. A number of applications can emerge from the ability to probe and control living matter on a large scale. In the short term, there two groups of application in the fields of sensing/diagnostic devices on one hand and bioproduction on the other. On the diagnostic front, molecular computing networks, either cell-free or cell-embedded, hold the promise of high diagnostic power in low volume and low energy consumption. The same can be said about biosensing. A sentinel engineered plant could monitor environmental pollution. Increasingly complex computing circuits could generate more information per unit of output (such as a change in color). DNA-based computing can be especially well suited for detection of other DNA or RNA inputs such as mutations.

Bioproduction. It is a consensus that bioproduction will eventually surpass traditional chemical engineering to become a major industry in 21st century. An important branch of bioproduction is metabolic engineering, the objective of which is to create cells that can effectively synthesize predesigned product, such as “green energy”, “green medicine”, and “green material”. Molecular computing has potential to become a key technology for control of metabolic pathways. Through internal sensing, computing and actuation of the metabolic regulatory networks, the embedded molecular computing device will direct the metabolic pathway to perform different tasks, such as growth, material production or apoptosis. It will optimize the desired product, giving the highest yield with the minimum energy consumption, and minimize the side-product to reduce environment burden.

Therapies. Therapeutic applications are very promising if not immediate-term anticipated results of molecular programming and computing. Any therapy represents, by definition, an attempt to control (human) physiology. Computing circuits, embedded within larger sensing/actuation networks, will form the basis for diverse family of therapeutic devices in the future. The key feature setting them apart from current therapies is the precise control over their mode of action, enabled by embedded information-processing circuits. Current state of the art prescribes two ways of building such systems. One is DNA nano-objects controlled by molecular networks integrating multiple inputs from the environment, operating e.g. in the bloodstream. Another solution is genetically-encoded systems. These can

be embedded in viral vectors and delivered to patient's cells, where they interface with cellular signals. Alternatively, they can be integrated in genetically-engineered cells that are infused in a patient and interact there with other cells and organs to achieve a desired effect. We envision that the two approaches (DNA/RNA/protein nano-systems and genetic encoding) will merge in the future into a unified technology.

A few possibilities arise from this concept.

- Control of timing and dosage of drug release/activation based on precise environmental and cellular cues. For example insulin can be released from engineered transplanted cells in a highly regulated fashion by an autonomously operating system without the need to perform injections.
- Specific therapeutic actuation only in cells that require it. This can be done by integrating multiple molecular markers indicative of a pathological state, in each and every cell. This concept can be implemented in cases where different cells exist in distinct states, such as cancer. It can also augment more traditional gene therapy approaches by precise control and restriction of a transgene's expression to a particular cell type. In either case, the efficacy of a treatment will increase dramatically with concomitant elimination of side effects.
- Multi-pronged *anticipatory* therapy. There exists increasing knowledge of the emergence of drug resistance in different disease. In certain leukemias, the response of a tumor to the first-line drug is usually stereotypical and reproducible. Complex therapeutic circuits could anticipate these chains of events, automatically triggering the most suitable treatment when sensing a transition from one disease state to the next.
- In the very long term, molecular computing in cells will enable highly sophisticated multi-cellular, multi-agent distributed systems that can comprise an artificial organ. A natural example of such an "organ" is the immune system, with many specialized cell types working together to achieve the desired effect. However, failure of the immune system allows the emergence of cancer or immune disorders. One can envision that an artificial multi-agent surveillance system will be built on top of the natural immune system to monitor it for possible failures and correct it in real time.

Gaps and challenges

In order for the above-mentioned milestones to be achieved, substantial progress must be made on a number of fronts, both in theoretical and experimental realms.

Challenges in theory development. While considerable advances have been achieved in molecular computing architecture, it is still not clear what is the best theoretical framework to guide the design of circuits compatible with living cells and organisms. Key features of such architectures should be robustness and reliability in real-life biological system. Here we are likely to benefit from learning from biology, because cells had to solve exactly the same problems over billions of years of evolution.

Linked to this challenge is the lack of reliable tools to simulate the operation of computing circuits in a living environment with a high degree of certainty. Finally, a robust technology requires automated computer-aided design (CAD) tools that will translate high-level specifications to working physical implementation. Such tools are by and large missing. In particular, no currently-available tool can simulate circuit operation with single-cell precision given physiological levels of heterogeneity and noise. Such single-cell CAD tools will be crucial to advance any medically-oriented application.

In this respect, the theoretical foundations for applying molecular computing in biology, biotechnology, and medicine is systems biology. Systems biology can help us to qualitatively understand the characteristics of biological regulatory networks, such as topological structure, dynamic behavior, functional properties, and their relations. However, at present systems biology cannot handle large biological systems, which are the main players in the application field. Often a quantitative modeling can only provide certain qualitative guidance and therefore must be improved.

Gaps in circuit construction. Molecular computing is a young field, and it yet has to discover the technology that will work in most cases (unlike in computer engineering, where VLSI represents a universal technological platform for almost all computing devices in the world). Nevertheless, a few front-runners have emerged, for example DNA strand displacement technology, DNA origami, transcriptional-, RNAi- and recombinase-based regulatory networks. DNA-based approaches have been shown to work in test tubes while synthetic regulatory circuits have been genetically-encoded and tested by and large only in cells. Given the large computational power of DNA systems, one challenge would be to adapt them to intracellular operation. Potential solutions could include the move from DNA to RNA building blocks (thus making them amenable to genetic encoding) or even developing peptide-based circuits that recapitulate some nucleic acids features. Another solution is to circumvent intracellular operation altogether and deliver

DNA-based networks directly into the bloodstream where they could interact with environmental cues or bind and effect specific cells through cell surface factors.

As far as traditional gene-circuit approaches are concerned, we are still far from having a genetic analog of a transistor – a simple universal building block that can be reused multiple times to enable potentially limitless complexity. Thus a concerted effort must be made to discover or invent such a unit.

Challenges in biological delivery. For any molecular computing system to function in cells or organisms, the components have to be delivered either across the cell membrane or across the body's defense mechanisms. There are multiple unanswered challenged in biological delivery, both with nucleic acid-based networks, with nano-objects, and with genetic material. Regarding the former, robust delivery methods are by and large missing. This represents a large gap that has to be filled if we wish to avoid genetic encodings and transitioning to RNA or peptide-based circuits. For genomic material, tools have been developed in the last few decades in the field of cell biology and gene therapy. Still most of these delivery methods are unsatisfactory in vivo (i.e., whole organism). More research is required to enable new materials for non-viral delivery. In this regard, 3D DNA origami might be a useful delivery approach as it allows condensing a large amount of DNA into very small particles. Regarding viral delivery, we need more efficient and safer viral vectors with increasing loading capacity. We emphasize that the real test of these vectors is in vivo. Finally, genome editing technologies need to advance to the point where large DNA payloads can be stably integrated in precise genomic locations with very high yield. This will enable application of genetically-encoded computing networks in the fields of immunotherapy, for example.

Interfacing with the host. Molecular computing circuits discussed in this section will be embedded in a biological host. Some of the interactions between the circuits and the host are desirable and in fact crucial for the intended operation. However, numerous non-specific interactions can also take place that may critically affect both the host and the computing circuits in unexpected ways. Such non-specific interactions have been overlooked so far, as the field has been addressing more pressing challenges. Among the questions are: How do cells interface with the engineered computing networks? How do complex DNA objects behave in vivo, in terms of immunogenicity, clearance, distribution and toxicity?

7. Applications in chemistry, physics, and materials

Programmed self-assembly. DNA nanotechnology, the design and self-assembly of artificial nucleic acid-based structures or systems, has developed with a breath-taking pace in recent years. The technology offers an unparalleled ability to control structure and function at the molecular level and the sizes of the structures are expanding towards the micrometer domain. Many intriguing structures ranging from 2D structures, solid brick structures, DNA boxes and spheres, tensegrity structures, various polyhedra, DNA structures with curvature and so forth have been made. Furthermore, several dynamic structures have been made, including walkers following designed tracks inspired by motor proteins such as dynein and kinesin. While these examples have demonstrated the structural and functional power of DNA origami, relatively few examples of implementation of DNA computing in such structures has been reported.

Programmed chemical synthesis and evolution. In nature the digital information of the genome is transcribed to RNA and then translated into proteins by the complex machinery of the ribosome. One of the holy grails of DNA computing and DNA nanotechnology is to make artificial systems that can convert digital information into chemical molecules with similar efficiency. The chemistry of such artificial systems may be extended to reactions other than amide formation and integrated with synthetic biology such systems might become artificial molecular factories.

In DNA-templated synthesis (DTS) the reaction between two or more molecules in solution is controlled, by modulating their local concentration via hybridization of the DNA strands to which the reactive molecules are linked, to a template or to each other. This approach mimics protein synthesis in cells, however the exquisite spatial control and catalytic effect in natural protein synthesis and thereby the efficiency of the synthesis is absent in DTS. Nevertheless, DTS has been applied to the formation of combinatorial libraries of molecules tagged with a DNA strands. Libraries of small molecules have been created and used for identification of small molecule binders to medically interesting targets (www.vipergen.dk) and for chemical reaction discovery, for example, in David Liu's group. A molecular assembly line system was furthermore created that allowed literally stepwise autonomous synthesis of a peptide like structure of a molecule on a DNA walker. However, DTS is extremely inefficient in terms of the amount of material that can be synthesized. Development of methods that would allow catalytic use of the controlling nucleic acid strands would be highly desirable. Alternatively, assembly of containers containing millions of reactant molecules may also be a solution to atom-efficient DTS synthesis of small molecules.

Transduction and interfacing. One approach to transduce a non-DNA signal to a DNA input signal is by conformal changes in a nucleic acid structures such as e.g. aptamers or riboswitches that can recognize and change conformation when

binding to a specific target. However, other molecular and biomolecular interactions resulting in a change in conformation of oligonucleotides, and thus leading to the generation of an input to a DNA computation, may also be envisioned. In nature the central outcome of signalling processes (signal transduction) is regulation of protein expression. In analogy to this it would be of major interest if the output of a DNA computation in an artificial DNA device would, through logical DNA operations, regulate the function of an enzyme that could exert its action to the environment or to a nanodevice. Such methods could enable the creation of autonomous multianalyte sensor systems that respond to the environment.

Integration of molecular components with silicon based electronics. A different aspect of transduction and interfacing is the interaction between individual molecules and CMOS electronics. As the size of DNA nanostructures increases towards the micrometer domain, and with improved strategies for immobilization of such structures on photo-lithographically patterned surfaces, it will eventually be possible to connect individual monodisperse DNA nanostructures to adjoining CMOS electrodes. This will eventually create a contact from the macroscopic world to individual molecules with a reliability and robustness that has not been accessible previously. This means that eventually *electronic* computation may be performed with molecular components integrated in CMOS electronics. Also, such constructs may provide an interface between DNA computing and electronic computing. A key challenge will be developing and characterizing robust and effective electronic connections between molecular components and between the molecular components to the macroscopic electrodes.

Nanorobots. Integration of information-processing capabilities with chemical and nanomechanical functions ultimately leads to the realization of nanorobotic systems. As their macroscale counterparts, such agents are able to sense their environment via appropriate sensor and transducer elements, perform calculations, and generate a response by which they can act back on their environment.

Nanorobots could assist in context-dependent assembly or chemical synthesis, they could function as intelligent, autonomous sensors ("molecular sentinels"), or controlled drug delivery units. First examples for the integration of information-processing and mechanical functions have been already demonstrated using self-assembled structures made from DNA. Landmark studies include molecular walkers that could be programmed to take specific routes on a network of molecular tracks, a molecular assembly line that could be programmed to generate a variety of different assemblies of nanoparticles, and molecular cages and containers that could open in response to external molecular signals.

Although such nanorobotic systems appear ultimately to be within reach, the challenges to their development are significant. Current systems are extremely slow relative to biological molecular motors, only the simplest nanorobotic systems can operate autonomously, and the range of actions they can perform (such as logic or picking up and carrying cargo) is extremely limited. Increased speed will require

improved scientific knowledge of macromolecular biophysics and design. Increased complexity will require integration of sensing functions, computation, mechanical and chemical actuation, all within a macromolecular scaffold such as DNA origami. Perhaps the toughest challenge will be providing a suitable energy supply for autonomously running systems, which requires the maintenance of non-equilibrium conditions.

Compartmentalized systems. One of the major differences between living and traditional chemical systems is compartmentalization. Spatial separation and structuring of molecules allows precise regulation of local production and degradation rates as well as control over molecular interactions and reactions. It is generally thought (and in some cases known) that spatial proximity and restricted diffusion leads to an increase of local concentrations, enhanced reaction fluxes, and a reduction of unwanted side-reactions and leaks. This increases the yield of reaction pathways, and allows the production of molecules that are not easily synthesized otherwise (famously, certain "biogenic" compounds are very hard to synthesize chemically). A variety of different approaches towards compartmentalization have been taken in the past – this includes encapsulation of reaction solutions in vesicles or microdroplets, lithographic patterning of molecules on surfaces (e.g., gene brushes), or microfluidic compartmentalization.

Apart from their role as simple "chemical reactors", compartmentalized chemical reaction networks could also function as autonomous agents themselves. Implementation of controlled exchange of materials and energy with the environment will be necessary to realize the vision of continuously operating, potentially self-regenerating or self-healing systems. Running molecular programs within these reaction compartments can result in autonomous, responsive, smart molecular systems. On the next level of organization this might lead to networks of interacting compartments with emergent properties resulting from cooperation or competition, and potentially even evolution.

To construct such an artificial cell will require compartmentalization of more complex reaction networks (molecular programs), and defining interfaces with the environment for nutrient supply and waste removal. Encapsulation of complex mixtures is physicochemical challenging—stability of membrane systems, maintenance of non-equilibrium conditions, transduction of information and energy and matter across membranes, are all potential roadblocks.

Programmed pattern formation. It is clear that – while being an extremely powerful basis for the realization of information-based molecular systems – passive self-assembly alone is not sufficient to generate certain types of large scale structures, and it is unable to create dynamically re-organizing, responsive, or context-dependent structures. One exciting possible application of molecular programming is the implementation of non-equilibrium pattern formation processes that also play important roles in biological morphogenesis and the development of organisms. In contrast to other chemical approaches towards

pattern formation, molecular programming enables the rational control of crucial physicochemical parameters for the generation of emergent behaviors, specifically rate constants, interaction strengths, diffusion coefficients. The control of active self-organization (rather than mere passive self-assembly) can thus provide an energy and resource-efficient route towards the generation of large and dynamic molecular structures.

First steps in this direction have already been taken. Several groups have generated programmable bistable systems, chemical oscillators, and filters that can lead to the generation of patterns when operated in a spatially extended (i.e., not well-mixed) context. On a different level, the formation of structure based on non-linear dynamical and reaction-diffusion effects has been studied extensively also in the context of synthetic biology, where the interaction of cells (typically bacteria) via diffusing signals, was used for the formation of structured ("artificially differentiated") biofilms.

Towards such "artificial developmental biology", next steps would be to learn how to "program" reaction-diffusion systems rather than well-mixed reaction systems, and to develop assembly strategies based on intelligent, interacting molecular building blocks (such as "swarms" of nanorobots). While the physical challenges here are significant, perhaps the toughest challenges will be conceptual: identifying effective models, abstractions, and ways to program (or even think about) pattern formation processes.

8. The role of technology

Molecular programming is the study of a new technology that is itself enabled by other technologies. For example, molecular programming has piggybacked on the development of relatively inexpensive synthesis of oligonucleotides for the biotechnology sector. More generally, molecular programming would not exist if it were not for availability of the of tools for manipulating DNA such as restriction enzymes, gel electrophoresis, fluorophore labeling of oligos, and a great number of other technologies originally designed for other purposes.

Molecular programming is now approaching the limits of these technologies and many in the field are seeking new and better technologies that enable, for example: the use of one thousand times more parts in a molecular circuit; programmed reactions that run at 100Hz; the creation of a universal constructor in the sense of von Neumann; and the application of molecular programming to cell-based characterization, diagnostics, therapeutics, and control.

The choice of which technology to explore is driven by several factors including what new engineering, science, or new applications the technology could enable.

In addition, new technologies that enable the development of new technologies for molecular programming are required at different levels of abstraction:

At the lowest level, new molecules, chemistries, and biological pathways amenable to programming are required. For example, the circuits constructed by Rondelez introduce a number of new biochemical components that dramatically improve the reliable construction of in vitro molecular circuits with topologies similar to those that work less well with other biochemistries, such as transcription-only circuits.

Each of these technologies requires a synthesis technology to make the parts available to molecular programmers. DNA synthesis seems to be improving dramatically in terms of length and quality. However, molecular programming still relies on enzymes made primarily for cloning. Efforts made to design new special purpose enzymes or cell-free systems are crucial.

At the next level are new gates or components, such as the seesaw gate of Qian and Winfree, that encapsulate and abstract a set of strand displacement reactions and enables circuits to be designed at higher levels of abstraction. Exploration into similar abstractions for nucleic acid circuits and other biochemistries is essential to enabling the design of more complex systems.

Finally, the use of lab automation such as liquid handling, lab-on-a-chip, or microfluidics as a means to more accurately assemble and characterize a greater number of molecular programs and variants is likely to speed the development of new molecular programming systems and paradigms.

As we increase the size and complexity of molecular programs, we will require substantially new ways to debug them. Currently molecular programming relies on fluorescently labeled probes to dynamically monitor systems, and gels to statically observe systems at different time points. With DNA origami and algorithmic self-assembly, the use of atomic force microscopes (AFMs) and transmission electron microscopes (TEMs) is standard and images of DNA assembly can be made. However, many other technologies can be brought to bear on the problem of observing and debugging molecular programs, such as other molecular labeling technologies, the use of vesicles to contain reactions, flow cytometry to measure activity of reactions in vesicles, high throughput methods, and sequencing to characterize for example inaccurately synthesized RNA products or circuits that were tuned through artifical evolution.

Much of the above discussion is based on molecular programming done with DNA (such as DNA origami and enzyme-free circuits) or DNA, RNA, and enzymes (such as genelets or the circuits developed by Rondelez). However, a number of other possible substrates for molecular computing are being developed as well.

Our understanding of protein folding and function lags our understanding of nucleic acids considerably. However, the ability to engineer proteins to specification has

begun to approach the complexity DNA self-assembly of simple shapes seen in the early 1990s. It is conceivable that new rules for designing and predicting the behavior of proteins will soon allow complex molecular programs to guide their dynamics, and possible to enable to behaviors and interactions with other living systems.

Finally, while synthetic biology in bacteria and other microorganisms has relied on the rewiring of found biological parts (e.g. transcription factors, receptors, or kinases), the last year has seen the development of entirely new libraries of orthogonal parts. For example, the reprogrammed CRISPR system allows for RNA mediated transcription of downstream genes and could allow molecular programmers to define predictable gene networks engineered to specification. Other transcription factor libraries are also becoming available, as well as other programmable mechanisms such as recombinases, receptors, and signaling systems.

9. Conclusions

Many, if not most, of the participants of the workshop felt that they were witnessing the birth of a new information technology. But despite its transformative potential, molecular programming and computing remains at an embryonic stage, with both enormous possibilities and enormous challenges. Practical applications are currently few and far between, and fundamental advances both in theoretical and experimental research will be necessary before the technology will bear fruit and live up to its promise. However, the path needed to get us there is clear—or, at least the general direction in which multiple paths must be explored to chart this novel territory. Let us review the picture presented in the previous sections.

Theory for molecular information systems. Theoretical models and insights have been essential to experimental advances in molecular programming. Often, experiments simply can't be conceived of until theory envisions the possibility. This is in contrast to fields where experimental discoveries lead, and theory follows to make sense of it. Thus, a thriving theory community must be an integral part of the molecular programming research field. Although a number of existing theoretical models have been well studied, better rigorous understanding of relationships between models—especially between models at higher and lower levels of abstraction—is needed. Furthermore, new models are needed to gain insights into how to program systems with complex, reconfigurable geometric arrangements, such as DNA origami, molecular robots, and developmental systems. Efficient methods to program such systems to carry out complex tasks, especially in the face of stochasticity and error-prone molecular processes, will require novel concepts as well as judicious application of known results from traditional fields within computer science. Importantly, molecular programming theory is unlikely to develop well in the absence of deep understanding of fundamental molecular

realities, and thus communication between experimentalists and theorists—and support of individuals who do both!—is extremely valuable at this stage of the field.

Modeling, analysis, specification, and design. The ultimate goal here is the development of high-level programming languages and associated compilers and design tools that automatically reduce a high-level description of a molecular system (such as a biochemical control circuit or a molecular robot) into sequences for DNA, RNA, and/or protein molecules that can synthetized in the laboratory. Doing so will require building an abstraction hierarchy (analogous to the one developed in electrical engineering and computer science to go from device physics to high-level languages like Mathematica) and the necessary software tools and theoretical foundations for proving the correctness of each stage of the compilation process. New concepts will be needed to incorporate probabilistic behavior, approximate models, and essential molecular aspects such as energy and material constraints, and spatial organization at all levels. As proof of principle, prototypes of such languages, abstraction hierarchies, and compilers have been developed and put to use for creating non-biological circuits and systems that operate within test tubes—but these prototypes languages are extremely limited in scope, have few or no theoretical guarantees, and to-date are unreliable in the sense that the resulting molecular systems must undergo several rounds of experimental debugging before they work. Beyond test-tube systems, creating programmable molecular systems that interact with, or operate within, biological cells is an important goal. Currently, the complexities and unknowns of biological cells make it unlikely that a fully specified programming language and fully automatic compiler can be developed in the near term. However, that puts the focus on the importance of developing accurate hierarchical and compositional models of whole organism that quantify uncertainty within the model, along with molecular programming languages that can provide guarantees of within-bounds behavior despite the uncertainty.

Applications in biology, biotechnology, and medicine. Molecular programming has great potential for transforming biosciences by embedding biochemical control circuits within biological and biochemical systems—much as the capability to embed general-purpose electronic control circuits within macroscopic electromechanical systems (everything from toaster ovens to automobile engines to manufacturing factories) has transformed today's most advanced technologies. Tomorrow's chemical information technologies will enable sophisticated diagnostics, biosensing, bioproduction, and therapies that sense their chemical environment, process chemical signals to extract information and make decisions, and produce chemical output. Rudimentary systems of this sort already exist, and they can be expected to steadily increase in complexity and capability. However, they are currently designed in an ad-hoc fashion; achieving the full potential of molecular programming will require "device-level" improvements, such as development of a biochemical mechanism analogous to the electronic transistor in the sense that the same basic device can wired up into an almost infinite variety of circuits. A number of candidates already exist, but it is not yet clear which, if any, will prove to be sufficiently reconfigurable and robust.

Applications in chemistry, physics, and materials. Beyond biological and biomedical applications, programmable molecular systems give us new ways to create complex non-biological molecules, systems, and materials. Already, techniques from structural DNA nanotechnology are letting us program molecular self-assembly to precisely organize hundreds to thousands of individual molecular components (such as aptamers, ribozymes, protein motors, enzymes, carbon nanotubes, metal nanoparticles, and quantum dots) according to nanometer-resolution 2D and 3D patterns. Such arrangements are impossible to achieve with other available techniques, and in combination with top-down lithographic methods, it may be possible to integrate nanometer-scale molecular components into CMOS electronic circuits. Dynamic DNA nanotechnology (reconfigurable structures, reaction cascades, and molecular motors) have already been used to programmably execute instructions to synthetize short polymers—first steps toward synthetic analogs of molecular machines such as the biological ribosome that reads information in RNA to synthesize the specified protein. More complex molecular manipulations could be performed (when molecular programming has advanced sufficiently) by teams of molecular robots.

The role of technology. Compared to more established engineering disciplines such as electrical engineering, where clean rooms with high precision instruments are necessary for device fabrication, molecular programming research has made relatively light use of technology—mostly relying on "off-the-shelf" commercial methods like mail-order DNA synthesis and relatively inexpensive analytical instruments. However, to bring molecular programming to the next level, a wider range of more advanced instruments and technologies will need to be employed, including lab automation such as liquid handling, lab-on-a-chip, and microfluidics, as well as high-throughput methods such as DNA sequencing, DNA chips, flow cytometry, encapsulation of reactions in micron-scale vesicles or droplets, and advanced microscopy methods.

Community. Science is done by people. Molecular programming will advance only in proportion to the capabilities of the people who make it happen. Because being at the forefront of molecular programming requires an aptitude for interdisciplinary research that spans chemistry, biology, physics, engineering, and computer science, it can be a challenge to find students who are motivated to and can master the relevant parts of each discipline. The field will have to solve this educational puzzle if it is to continue to grow.

It is difficult to know when molecular programming will become a mature technology. But already it is an exciting frontier where many disciplines meet to make possible molecular engineering of a sophistication that was barely imaginable even a decade ago.

10. Appendix: program committee, participants, and program

Program committee

Mitra Basu	NSF, USA
Yaakov Benenson	Biosystems Science & Eng., ETH Zurich at Basel, Switzerland
Roumen Borissov	CORDIS FET
Luca Cardelli	Microsoft Research Cambridge, UK
Kurt Gothelf	Chemistry, Aarhus University, Denmark
Manoj Gopalkrishnan	Computer Science, Tata Institute, Mumbai, India
Masami Hagiya	Computer Science, University of Tokyo, Japan
Christian Joachim	CNRS, Toulouse, France
Eric Klavins	Electrical Engineering, Univ. of Washington, Seattle, USA
Marta Kwiatkowska	Computer Science, University of Oxford, UK
Satoshi Murata	Bioengineering & Robotics, Tohoku University, Sendai, Japan
Qi Ouyang	Physics, Peking University, Beijing, China
Grzegorz Rozenberg	Computer Science, University of Leiden, The Netherlands
Ehud Shapiro	Computer Science, Weizmann Institute, Rehovot, Israel
Friedrich Simmel	Physics, Technical University of Munich, Germany
Mukund Thattai	National Center for Biological Science, Bangalore, India
Ron Weiss	EECS and Biological Engineering, MIT, Cambridge, USA
Erik Winfree	Computer Science & Bioengineering, Caltech, Pasadena, USA

The Danish Council for Strategic Research

Advances in Molecular Programming and Computing: Toward Chemistry as a New Information Technology

Workshop 2 – 4 May 2013, Copenhagen

Participants – in alphabetic order

Ebbe Andersen, Assistant Professor, Aarhus University
esa@inano.au.dk

Henrik Reif Andersen, CEO, ConfigIT (Chairman, Danish Council for Strategic Research – Programme Commission on Strategic Growth Technologies)

Roy Bar-Ziv, Associate Professor, Weizmann Institute of Science
roy.bar-ziv@weizmann.ac.il

Mitra Basu, Program Director, National Science Foundation
mbasu@nsf.gov

Yaakov Benenson, Professor, ETH Zurich
kobi.benenson@bsse.ethz.ch

Luca Cardelli, Computer Scientist, Microsoft Research
luca@microsoft.com

Anne Condon, Professor and Head, University of British Columbia
condon@cs.ubc.ca

Andrew Ellington, Research Professor of Biochemistry, University of Texas at Austin
andy.ellington@mail.utexas.edu

Elisenda Feliu, Postdoc, University of Copenhagen
efeliu@math.ku.dk

Harold Fellermann, Postdoctoral Research Fellow, University of Southern Denmark
harold@sdu.dk

Ulrich Gerland, University Professor, LMU München
gerland@lmu.de

Manoj Gopalkrishnan, Reader, Tata Institute of Fundamental Research
manoj.gopalkrishnan@gmail.com

Kurt Vesterager Gothelf, Professor, Aarhus University
kvg@chem.au.dk

Masami Hagiya, Professor, University of Tokyo
hagiya@is.s.u-tokyo.ac.jp

Natasa Jonoska, Professor, University of South Florida
jonoska@mail.usf.edu

Eric Klavins, Associate Professor, University of Washington
klavins@uw.edu

Yamuna Krishnan, Reader F, National Centre for Biological Sciences
yamuna@ncbs.res.in

Marta Kwiatkowska, Professor, University of Oxford
jordan.summers@cs.ox.ac.uk

Timothy Lu, Assistant Professor, MIT
timlu@mit.edu

John McCaskill, Research Group Leader, Ruhr-Universitaet Bochum
john.mccaskill@rub.de

Satoshi Murata, Professor, Tohoku University
murata@molbot.mech.tohoku.ac.jp

Carlos Olguin, Head of Bio/Nano/Programmable Matter Group, Autodesk, Inc.
carlos.olguin@autodesk.com

Qi Ouyang, Professor, Peking University
qi@pku.edu.cn

Susan Peacock, ICT Portfolio Manager, EPSRC
susan.peacock@epsrc.ac.uk

Jean Peccoud, Associate Professor, Virginia Bioinformatics Institute at Virginia Tech
jlewis@vbi.vt.edu

Lise Refstrup Linnebjerg Pedersen, Centre Manager, PhD, Aarhus University
lrlp@chem.au.dk

Andrew Phillips, Scientist, Microsoft Research
andrew.phillips@microsoft.com

Lulu Qian, California Institute of Technology
luluqian@caltech.edu

John Reif, Professor, Duke University
reif@cs.duke.edu

Yannick Rondelez, Researcher, CNRS-IIS
rondelez@iis.u-tokyo.ac.jp

Rahul Sarpeshkar, Associate Professor, MIT
rahuls@mit.edu

Ehud Shapiro, Professor, Weizmann Institute
ehud.shapiro@weizmann.ac.il

Friedrich Simmel, Professor, TU München
simmel@tum.de

David Soloveichik, Center for Systems and Synth Bio Fellow, University of California, San Francisco
David.Soloveichik@ucsf.edu

Brynne Stanton, Massachusetts Institute of Technology
stantonb@mit.edu

Milan Stojanovic, Columbia University
mns18@columbia.edu

Chris Thachuk, Research Assistant, University of Oxford
chris.thachuk@cs.ox.ac.uk

Andrew Turberfield, Professor, University of Oxford
a.turberfield@physics.ox.ac.uk

Gregory Wallraff, Staff Scientist, IBM
gmwall@us.ibm.com

Jessica Walter, Postdoc, UCSF
jessica.walter@ucsf.edu

Erik Winfree, Professor, Caltech
winfree@caltech.edu

Carsten Wiuf, Professor, University of Copenhagen
wiuf@math.ku.dk

Staff – Danish Council for Strategic Research

Birthe Schouby
Jimi Walldo Rasmussen
Mathias Vinzent-Kerkov
Maibrit Kanding

The Danish Council for Strategic Research

Advances in Molecular Programming and Computing: Toward Chemistry as a New Information Technology

Workshop 2 – 4 May 2013, Copenhagen

Program

Thursday 2 May: Opening and lectures at Carlsberg Academy

13:00 – 13:30 Opening: chaired by Kurt Gothelf

Welcome and introductory remarks by:

- Henrik Reif Andersen, Danish Council for Strategic Research
- Carlsberg Academy – a short introduction to the history of the buildings
- Erik Winfree, Caltech, Chairman of the Program Committee

13:30 – 15:00 Session 1: chaired by Luca Cardelli

13:30 – 14:00 KEYNOTE: Anne Condon, "On the usefulness of models for biomolecular programming"

14:00 – 14:20 David Soloveichik, "Strand displacement cascades and the notion of a chemical algorithm"

14:20 – 14:40 Andrew Phillips, "Programming DNA strand displacement systems"

14:40 – 15:00 Lulu Qian, "Molecular programming with artificial nucleic acid systems"

15:00 – 15:20 Break with coffee

15:20 – 16:40 Session 2: chaired by Kobi Benenson

15:20 – 15:40 Tim Lu, "Scalable platforms for computation in living cells""

15:40 – 16:00 Rahul Sarpeshkar, "Analog synthetic biology: from cells to electronics and electronics to cells"

16:00 – 16:20 Brynne Stanton, "A genetic language to build extensible programs in microbial and mammalian cells"

16:20 – 16:40 Jean Peccoud, "Defining domain-specific DNA programming languages"

16:40 – 17:00 Break

17:00 – 18:30 Session 3: chaired by Fritz Simmel

17:00 – 17:30 KEYNOTE: Andy Ellington, "The unique programmability of nucleic acids"

17:30 – 17:50 Yannick Rondelez, "Rational design of temporal and spatiotemporal emergence"

17:50 – 18:10 Roy Bar-Ziv, "Cell-free protein synthesis and assembly on a biochip"

18:10 – 18:30 John McCaskill, "Towards DNA-assembled electronic chemical microprocessors"

Evening activities

19:00 Dinner at Carlsberg Museum

Approx. 21:00 Joint transport to Magleaas Conference Center

Approx. 21:45 Arrival at Magleaas Conference Center and Lodging

Friday 3 May: Lectures and group discussions at Magleaas Conference Center

08:00 – 09:00 Breakfast

09:00 – 10:30 Session 4: chaired by Eric Klavins

09:00 – 09:30 KEYNOTE: Andrew Turberfield, "Molecular machinery from DNA"

09:30 – 09:50 Yamuna Krishnan, "Designer nucleic acids to probe and program the cell"

09:50 – 10:10 Jessica Walter, "Designing and programming spatial organization in cells"

10:10 – 10:30 Milan Stojanovic, "Progress in DNA computing and sensing"

10:30 – 10:50 Break with coffee

10:50 – 11:50 Session 5: chaired by Marta Kwiatkowska

10:50 – 11:10 Ulrich Gerland, "About sequential logic in biomolecular information systems"

11:10 – 11:30 Elisenda Feliu, "Constraining the parameter region of chemical reaction networks for switch-like behaviour"

11:30 – 11:50 Harold Fellermann, "Integrating molecular production and information processing in an artificial sub cellular matrix"

12:00 – 13:30 Lunch and photo

13:30 – 15:00 Session 6: chaired by Satoshi Murata

13:30 – 14:00 KEYNOTE: John Reif, "Future Challenges for DNA-Based Nano-Architectures and Nano-Devices"

14:00 – 14:20 Natasha Jonoska, "Recursion and spatial aperiodic arrangements in molecular self-assembly"

14:20 – 14:40 Ebbe Sloth Andersen, "Construction of biomolecular architectures through computer-aided design"

14:40 – 15:00 Break with coffee

15:00 – 18:30 Parallel working group discussions

15:00 – 15:30 Description of the task: Mitra Basu and Erik Winfree on overall goals
Each working group leader describes what they hope to address (six 3-minute talks)

15:30 – 18:30 Discussions in working groups (primary discussion leader noted)

- Room 1: Theory for molecular information systems (Manoj Gopalkrishnan)
- Room 2: Modeling, analysis, and specification (Marta Kwiatkowska)
- Room 3: System design (Satoshi Murata)
- Room 4: Architectures and applications in biology & medicine (Kobi Benenson)
- Room 5: Architectures and applications in chemistry, physics, & materials (Fritz Simmel)
- Room 6: Technology (Eric Klavins)

Evening activities

19:00 Dinner
Informal discussion

Saturday 4 May: Discussions in plenum and wrap-up at Magleaas

08:00 – 09:00 ***Breakfast***

09:00 – 12:00 ***Working group presentations & discussion, chaired by Udi Shapiro***
Each working group leader summarizes their assessment (10 minutes)
Followed by discussion (15 minutes)

09:00 – 10:15 Groups 1, 2 and 3

10:15 – 10:45 Coffee break

10:45 – 12:00 Groups 4, 5 and 6

12:00 – 13:00 ***Lunch***

13:00 ***Departure for participants that are not program committee members***

13:00 – 16:00 Report writing by the program committee.
Each working group leader, with help, prepares written report on their assessment

16:00 ***Departure for program committee members***

Members of the program committee and agency representatives:
Mitra Basu, Kobi Benenson, Luca Cardelli, Manoj Gopalkrishnan, Kurt Gothelf, Masami Hagiya, Eric Klavins, Marta Kwiatkowska, Susan Peacock, Satoshi Murata, Qi Ouyang, Udi Shapiro, Fritz Simmel, Erik Winfree, Birthe Schouby, Mathias Vinzent-Kerkov