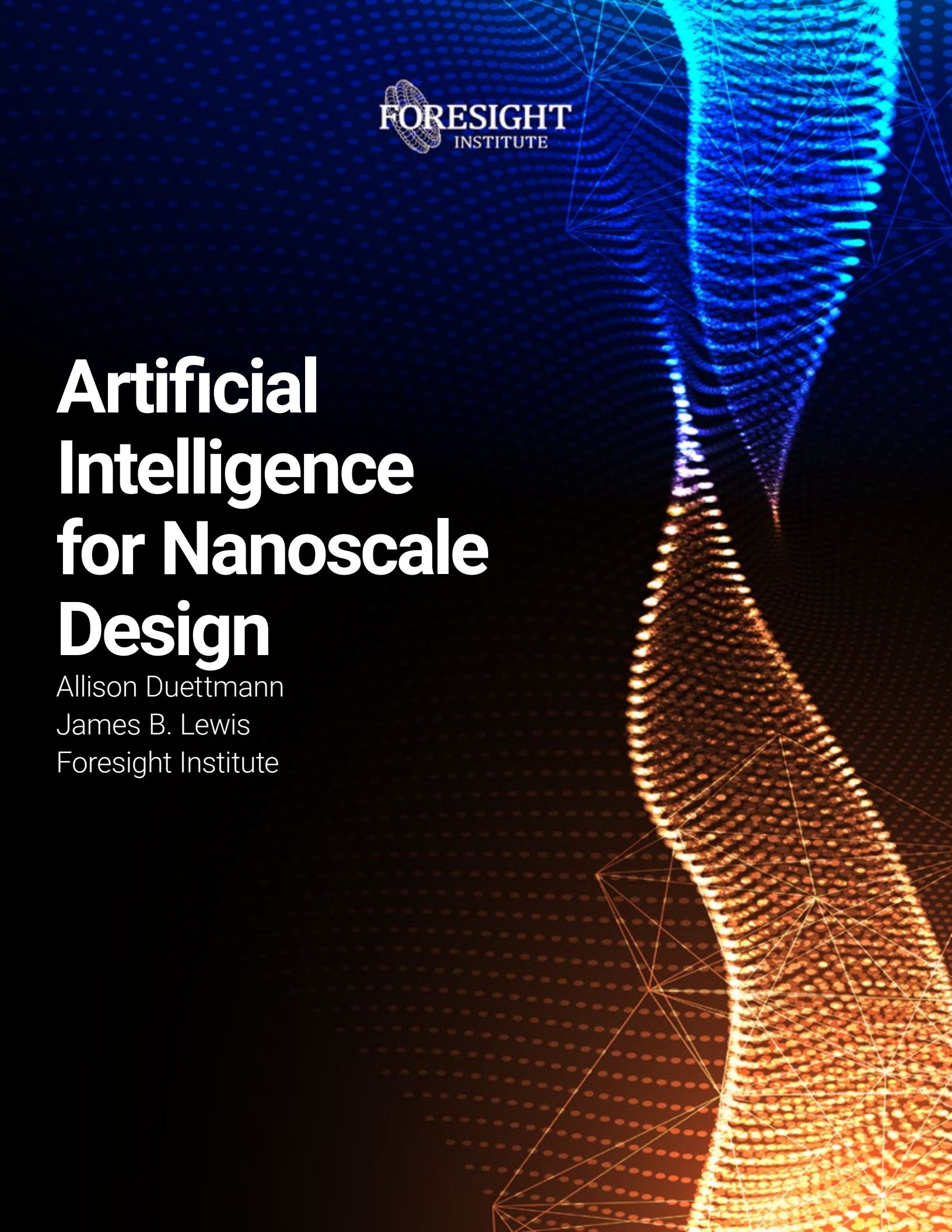


Artificial Intelligence for Nanoscale Design

Allison Duettmann
James B. Lewis
Foresight Institute

Artificial Intelligence for Nanoscale Design

A proposal sketch based on the Foresight Institute Research Workshop series

Allison Duettmann
James B. Lewis
Foresight Institute

Participants

Monica Anderson, Syntience Inc.

Jonathan Barnes, Dept. of Chemistry, Washington Univ.

Jeremy Barton, Nanofactory Corporation

Chuyang Cheng, Dept. of Chemistry, Northwestern Univ.

Martin Edelstein, Covalent/ Agua Via

William A Goddard, III, Caltech, Dept. of Chemistry, Materials Science, and Applied Physics

Steve Fowkes, Nanopolymer Systems

Ben Goertzel, Hanson Robotics, Aidyia Limited, AGI Society and OpenCog Foundation

J. Storrs Hall, Scientist and Author

Sergei V. Kalinin, Center for Nanophase Materials Sciences, Oak Ridge National Lab

Kent Kemmish, Stealth Biotech

Tarik Koc, OccamzRazor.com

John Koza, Computer Science, Stanford Univ.

Markus Krummenacker, SRI International

James B. Lewis, Foresight Institute

Joseph W. Lyding, Dept. of Electrical & Computer Eng., Univ. of Illinois-Urbana

Clinton (Cosmo) Mielke, Project Infinome

Gayle Pergamit, Covalent/Agua Via

Kutay Sezginal, Univ. of Pittsburgh

Christian Schafmeister, Dept. of Chemistry, Temple Univ.

Eva-Maria Strauch, Dept. of Biochemistry, Univ. of Washington

Berhane Temelso, Computational Chemistry, Furman Univ.

Katharina Sophia Volz, OccamzRazor

Christopher Wilmer, Chemical and Petroleum Engineering, Univ. of Pittsburgh

Zhiyong Zhang, Dept. of Computational Science, Stanford Univ.

Table of Contents

Foreword

Foresight Institute is a non-profit organization focused on supporting the benefits and avoiding the downsides of technologies of fundamental importance for the human future, especially molecular machine nanotechnology, cybersecurity, and artificial intelligence. In addition to awarding the Feynman Prize and sponsoring Foresight Fellowships, Foresight hosts invitational workshops to selectively advance beneficial technologies, researches the risks associated with those technologies, and hosts meetings to support our flourishing using those technologies.

As became apparent in The Foresight Institute research workshop series, many obstacles to advancing individual scientific disciplines that resurfaced repeatedly throughout different workshops appear to be solvable using AI tools. To directly address this opportunity, two workshops, AI for Scientific Progress (Sep. 30-Oct. 2, 2016) and AI for Atomic Precision (May 27-28, 2017), were held. AI for Nanoscale Design is the final product of these workshops: An earlier version of the proposal won both the Jury Award and the Public Choice Award at the AI for Scientific Progress workshop. Following this success, an initial proposal draft for NanoMind was submitted by Ben Goertzel to Foresight Institute, and the second workshop, AI for Atomic Precision, was initiated to continue work on the research proposal.

AI for Nanoscale Design is an extension of Christian Schafmeister's CANDO software to incorporate genetic programming and OpenCog's framework, as well as a greater variety of simulations, platforms and other external AI tools. The goal is to create a unified software environment which eliminates the need to learn and use different tools manually. Instead AI for Nanoscale Design allows easy and unified use of different simulators, and different AI algorithms that serve different functions, such as design or accelerate a simulator.

Foreword

AI for Nanoscale Design combines three main building blocks:

1) An advanced software framework for AI-powered nanoscale design, which is proposed in the early draft of the NanoMind proposal
2) the CANDO framework, an existing software tool for chemistry, which is proposed as technical foundation for NanoMind
3) Genetic Programming, an existing machine learning technique, which is proposed as a potential AI tool to expand on existing software solutions to chemistry

This white paper is divided into four sections, allowing the reader to escalate depending on expertise and interest.

Part 1: AI for Nanoscale Design — for those only interested in the research proposal
Part 2: Background information — for those interested in more detailed information on the building blocks of NanoMind
Part 3: Introduction to AI and Molecular Machines — for those interested in a primer on the scientific fields addressed by NanoMind
Part 4: Bonus Material — for those interested in workshop discussion on how artificial intelligence could facilitate the development of atomically precise nanotechnology

Foresight Institute would like to thank participants of both workshops for their creativity, commitment, and collaborativeness that made the generation of this proposal sketch possible.

For more information on the proposal, potential next steps, our workshops, or Foresight Institute, please email foresight@foresight.org.

Allison Duettmann
Foresight Institute

Part 1: A Research Proposal for AI for Nanoscale Design

AI for Nanoscale Design is based on the observation that many of the challenges that workshop participants in the Foresight Institute research workshop series repeatedly reported facing appear to be solvable by AI-powered tools. See below for a wishlist of tools, generated by participants throughout several workshops:

- *Software for molecule design*: Software that allows one to design molecules, explore different functions of the molecules, and search through millions of designs quickly. Gazebo is an example from robotics: It is a robot simulator with models for each motor, gearbox, etc. To simulate a design, *e.g.*, for a walking robot, one can piece together the type of desired robot and start with any AI algorithm to run the simulator. In chemistry, a broad range of expert knowledge about what properties molecules should have is readily available and could be used as soft constraints in the fitness function of advanced algorithms. A problem with current solutions is the integration of different software and programming languages, *e.g.*, combining high-level chemistry software that knows Prelog rules and builds 3-dimensional chemical structures with more dynamic languages where one can do programming.

- *Predicting Structure from a Sequence of Monomers*: When combining a sequence of monomers that build a structure, it would be useful to create different prototypes or models to predict how these structures are going to perform once they are assembled, e.g., for polymers with metal atoms in them, where the metal atoms are and what kind of proximities there will be. For more on the "sequence of monomers that build a structure", see Foresight's 2016 Atomic Precision for Energy Workshop pp. 11-19.

- *Assembling Functional Scaffolds or Surfaces*: From an engineering point of view, analyzing a molecule is not enough if one wants to assemble it on a functional scaffold. Different assembling strategies are available, *e.g.*, using DNA origami, scanning probes, or manipulations using electron beams. A system that can learn from experimental examples, and develop the theoretical tools that allow one to predict how molecules change under the action of specific kinds of manipulation, would be desirable for gaining knowledge of how to assemble them and how to edit any molecule without having to do cost and time expensive lab experiments. From an experimental viewpoint, it would be valuable to simulate how a chemical reaction

translates to a specific surface so that one can use a molecule on a surface to cause a specific chemical reaction.

■ *Lack of Molecular Models*: Nanotechnology lacks models—at the molecular level, most models are borrowed from physics, chemistry, and models of mechanics at the atomic level. However, nanotechnologists generally do have a treasure trove of data. They could provide data, *e.g.*, spectroscopy data of molecules and X-ray crystallography data, but need guidance by AI researchers on what molecular parameters are required to design the system.

■ *Lack of Consistent Database for Molecular Machines*: When studying molecular machines, structures and 3D coordinates of molecules are not well-validated, making it difficult and time-consuming to draw the data manually. Having a tool that could generate a structure by reading descriptions of the molecules would allow detailed study of the systems, including study of the motions of these machines in a simpler way. This database could be the base for a larger database addressing how different systems integrate on different scales, which is useful in building systems of molecular machines.

■ *Bridging Top-Down and Bottom-Up Data*: In biology, top-down approaches have increasingly large databases, e.g., of protein localization data within cells, while bottom-up approaches deliver X-ray crystallographic data, giving atomic coordinates within crystal structures. Unification of bottom-up and top-down is needed, e.g., what protein complexes might look like based on binding data or pathway data.

■ *Relationships Among Molecular Structures:* It would be useful to use the vast amount of scientific data in chemistry to understand the role of molecules, the cause and effect relationships among them, and how to extract those to create valuable applications to generate a more universal view of what is happening.

■ *Detailed Search for Protein Design & Ligand Binding:* There are some examples of design results[1] for designing a protein that would fit a specific shape to within a few nanometers, *e.g.*, when starting with an amino acid sequence. However, for other problems, *e.g.*, ligand binding, finding and integrating data is much harder. At Foresight's workshop on AI for Scientific Progress, it was noted that one research organization has 125 billion binders for a single protein, based on DNA-coded computational chemistry. While it is useful to have a single well-defined target and 125 billion variants that bind to it, the variants might not bind to the same site on the protein. A crystal structure gives one all the residues, structure, and where the ligand binds, but for each ligand the residues will have different conformations. The crystal structure bears no information on the conformations of the residues that interact with the ligand to stabilize the binding. There is software that takes a pharmacophore and goes through a couple million molecules to find the thousand or so that are best, narrowing the search field from two million to a thousand molecules. In those thousand there could be ten or twenty good binders, but the software is silent on which thousand are the good ones because it cannot identify conformations that match the pharmacophore. The problem is to distill, from those thousand, the one or two that might be nanomolar binding. Currently, researchers can handle the quality of the force field, and how the residue side chain conformations change, but the calculations are not systematized well enough to trust a program to give a reliable answer.

■ *Problem of Overfitting Data*: When applying machine learning to genomics, the required data depends on the combination of genes and the complexity of the phenotype. In the case of Alzheimer's disease or Parkinson's disease, useful discoveries are possible with 500 to 2000 cases and an equal number of controls.

[1] For a recent review: "The coming of age of de novo protein design" Po-Ssu Huang, Scott E. Boyken & David Baker. Nature 537, 320–327 (15 September 2016). Full text PDF available courtesy of the Baker Lab

However, in many cases it is a particular challenge to not overfit the data: one sets aside ¾ of the data, then cross validates and so forth.

While some of the above challenges are ambitious to solve, AI for Nanoscale Design proposes a starting point to address many of the challenges. The next section discusses the proposal in more detail.

AI for Nanoscale Design Proposal Presentation

By Ben Goertzel, OpenCog Foundation and Christian Schafmeister, Temple University

The video of the project presentation can be viewed here. The transcript of the presentation is below.

Ben Goertzel: I am going to give a brief introduction and overview to some thinking we have been doing regarding application of AI to nano-scale design, and then I will turn over to Christian Schafmeister to go into more technical details on both the software infrastructure and the molecular design side of things.

I myself am an AI researcher. Among other things I am leading an open source AI software project called OpenCog, which I am not going to tell you too much about, but it is a neuro symbolic system using a weighted, labelled hypergraph to represent knowledge. With OpenCog we are doing a whole bunch of different applications. One application is controlling humanoid robots to make realistic facial expressions, to recognize emotions in people's faces, and to hold natural language dialogs, and so forth, which is being done in our lab in Hong Kong.

Another application I have been doing, which moves a little closer to the event at the moment, is applying OpenCog to analyze biology data. We have been analyzing SNP (single-nucleotide-polymorphism) data, gene expression data, and various other types of data. In one example, OpenCog looked at a bunch of SNP data related to supercentenarians, people living 110 years or over, and it found a pattern. If they have a SNP here or a SNP here and a SNP there and no SNP there, and so forth, then they are more likely to live a long time. This is AI applied to data analysis.

Having applied AI in a bunch of different practical domains, as well as working toward the longer-term goal of Artificial General Intelligence—making AIs that can really think like people—it often occurred to me that it would be interesting to apply AI in the domain of nanoscale design. Of course, you could say, Let's just build a human-level thinking machine first, and let that human-level AGI solve all the other problems. That may be the most rational course, but we don't know how long it will take to get to a human-level AI that can really solve all of our problems for us. It is also a lot of fun to think about solving near-term problems with artificial intelligence.

I knew that AI had been used in many design problems before, which we will talk about in a few minutes, so to design things at the nanoscale seemed like a problem that would be very appealing to do with AI systems, for a number of reasons. First, it is not the kind of thing that humans really evolved to do. Recognizing faces, which AI

has succeeded to do well lately, is something that evolution worked pretty hard to get us to excel at, because it is very valuable to us as human beings. On the other hand, we do not have much inductive bias evolved into our brains for figuring out how a protein folds, or how to position metal atoms on the inside of a tube. That is not something that humans are terribly good at. AI may be biased to other domains, and thus may be much better than humans at thinking about nanosystems.

The kind of system that I started thinking about is, suppose a human being gives some constraints and requirements on the kind of nanosystem to build: what molecules or atoms are to be put together, what is the whole system to do? Then suppose the AI explores the design space to find a system within those constraints that will fulfill the given requirements. One thing you could do in principle is to connect the AI to laboratory equipment so it can experiment with designs as it manipulates them. Of course that is not only cumbersome in practice but also expensive to do. What we want to do in practice is connect the AI system to simulation engines, and there are multiple simulation engines to simulate different systems at different scales and in different ways. There is also an interesting line of research where you use AI to accelerate the simulation process by having an AI simulate the simulator, using a deep neural network. That is a new area of research, but in some cases it has been shown that, after some training, an AI can do what a simulator does much faster. Of course, AI can also be useful to analyze data coming out of a simulator or lab experiment.

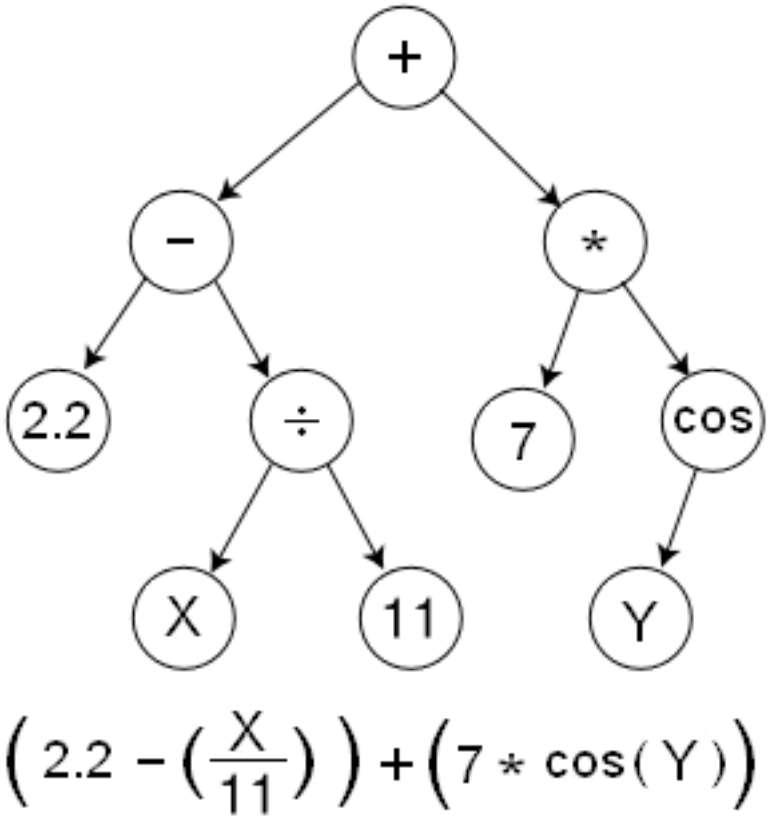
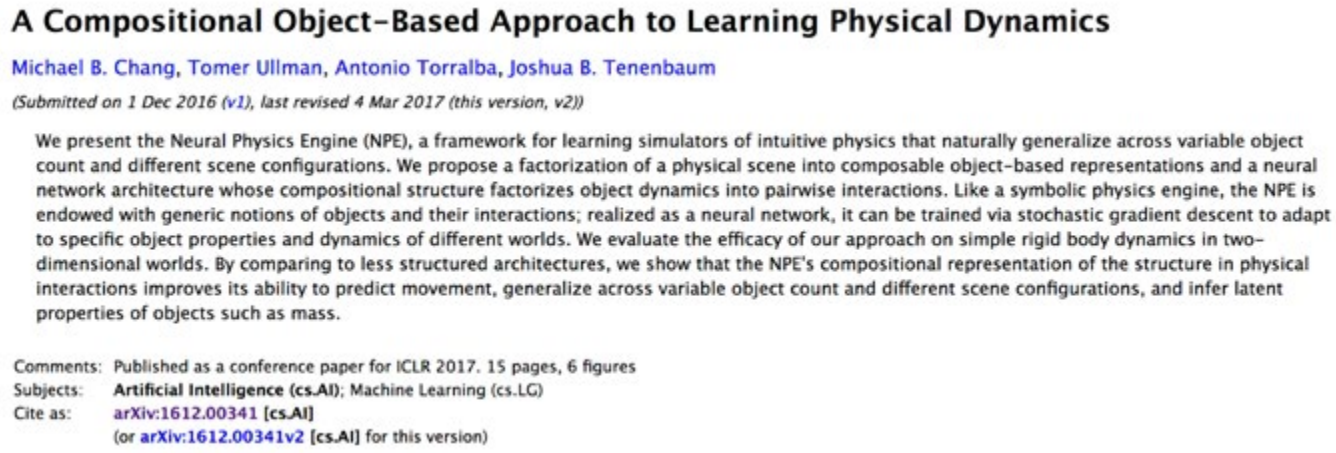


Figure 1. *Genetic Program Tree. A function represented as a tree structure.* Credit: Wikipedia. This work has been released into the public domain by its author, BAxelrod at English Wikipedia. This applies worldwide.

So what we are looking at here, is how would you use various AI algorithms, in various different roles, integrated together within a coherent software platform to accelerate and improve nanoscale design. There are a number of types of AI that could be used. I am going to quickly mention three of them before turning things over to Christian.

One of them is called Genetic Programming (see section Genetic Programming). Genetic Programming is a method of automated program learning, which emulates the process of evolution by natural selection. Usually what is meant by a program in genetic programming is a little tree (Fig. 1). In this case, a program with +, -, x, numbers, etc. simply evaluates this algebraic expression. You can also have programs that contain operations. This is the end of a bond; we are inserting something here; we are adding something there. You have a program that builds some structure as you execute it. Christian will talk about some pertinent examples of that.



A Compositional Object-Based Approach to Learning Physical Dynamics

Michael B. Chang, Tomer Ullman, Antonio Torralba, Joshua B. Tenenbaum

(Submitted on 1 Dec 2016 (v1), last revised 4 Mar 2017 (this version, v2))

We present the Neural Physics Engine (NPE), a framework for learning simulators of intuitive physics that naturally generalize across variable object count and different scene configurations. We propose a factorization of a physical scene into composable object-based representations and a neural network architecture whose compositional structure factorizes object dynamics into pairwise interactions. Like a symbolic physics engine, the NPE is endowed with generic notions of objects and their interactions; realized as a neural network, it can be trained via stochastic gradient descent to adapt to specific object properties and dynamics of different worlds. We evaluate the efficacy of our approach on simple rigid body dynamics in two-dimensional worlds. By comparing to less structured architectures, we show that the NPE's compositional representation of the structure in physical interactions improves its ability to predict movement, generalize across variable object count and different scene configurations, and infer latent properties of objects such as mass.

Comments: Published as a conference paper for ICLR 2017. 15 pages, 6 figures
Subjects: **Artificial Intelligence (cs.AI)**; Machine Learning (cs.LG)
Cite as: arXiv:1612.00341 **[cs.AI]**
(or arXiv:1612.00341v2 **[cs.AI]** for this version)

Figure 2. Neural Nets to Accelerate Physics Simulation. Credit: Michael B. Chang et al. The above image is the abstract of the paper "A Compositional Object-Based Approach to Learning Physical Dynamics" by Michael B. Chang, Tomer Ullman, Antonio Torralba, Joshua B. Tenenbaum, published in arXiv, accessed August 17, 2018.

In general, whatever program you want to learn, the genetic programming process works by taking a population of possible programs, evaluating how well they work at doing something, selecting among the ones that did well, mutating them, crossing them over, and maybe doing some other operations. This gives you more programs in your population. Similarly to evolution by natural selection, you are trying to evolve new programs. This particular type of AI has proved very good at designing various systems, so I think it is highly relevant to nanoscale design, as Christian will go through in more detail.

In a different application of AI that fits in the same framework, there has been recent work using neural networks to accelerate physics simulation (Fig. 2). Joshua Tenenbaum and his colleagues made something called the Neural Physics Engine, where they take a physics engine as used in a game, Newtonian mechanics on solid objects, and they show that by using a neural network you can do the same calculations that the physics engine does, but much faster. In essence, the neural net figures out, by studying many, many different simulated scenarios, for each little configuration of solid objects in the simulation, what is the most likely future of a configuration of solid objects (Fig. 3). Then it can in effect jump ahead many steps in the physics engine, so the neural net can do faster simulation than the game engine. Applying that to the more complex simulations involved in nanotechnology is certainly not trivial, but it is the same formal problem. This research was just published at the end of last year, so this is hot-off-the-presses stuff[2].

[2] "A Compositional Object-Based Approach to Learning Physical Dynamics" MB Chang, T Ullman, A Torralba, JB Tenenbaum. arXiv:1612.00341v2
[cs.AI] for this version) [v2] Sat, 4 Mar 2017 17:44:06 GMT (6899kb,D) https://arxiv.org/abs/1612.00341

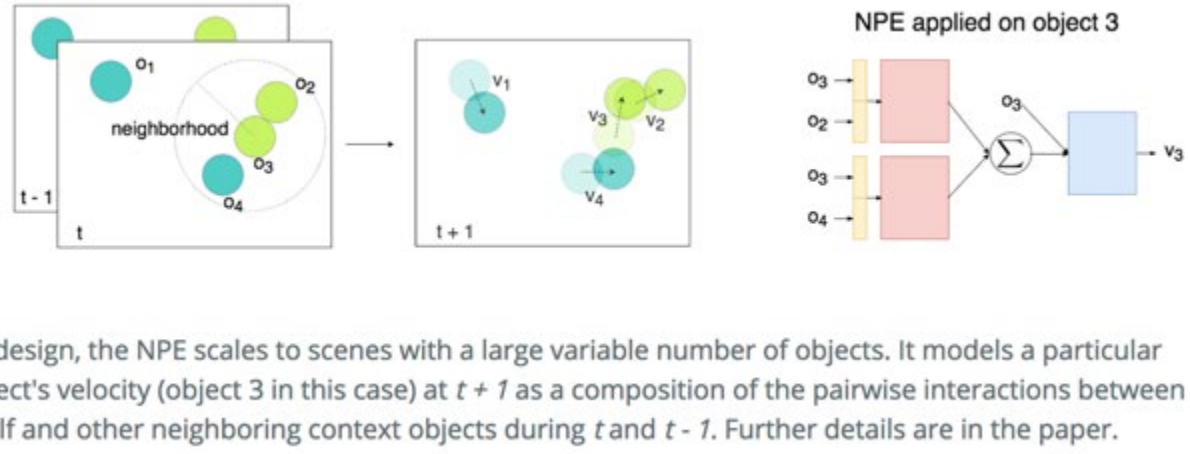


Figure 3. *Neural Nets to Accelerate Physics Simulation.* Credit: The github page for "A Compositional Object-Based Approach to Learning Physical Dynamics" by Michael B. Chang, Tomer Ullman, Antonio Torralba, Joshua B. Tenenbaum, accessed August 17, 2018. "We propose a factorization of a physical scene into composable object-based representations and a neural network architecture whose compositional structure factorizes object dynamics into pairwise interactions." This figure is from the github page, which presents "results that show the NPE's efficacy in prediction, generalization, and inference."

Another area, which is something that I have worked on a lot, is application of automated reasoning (Fig. 4). We have a lot of equations in chemistry and physics, a lot of formalized knowledge. You can use AI to reason on that formalized knowledge to draw new inferences, new hypothetical conclusions. This hypothetical knowledge can be used to guide evolutionary design. These different types of AI can all work together.

On the synthesis of machine learning and automated reasoning for an artificial synthetic organic chemist

Check for updates

Maneesh K. Yadav[a]

Author affiliations

Abstract

This perspective outlines current capabilities and limitations of state-of-the-art artificial intelligence methods as applied to automating the planning of synthetic routes in organic chemistry. Synthetic organic chemistry is viewed from the perspective of two prominent approaches: deep neural networks and SAT-solver based automated reasoning. After introducing these concepts to non-computer scientists, the expected performance of these approaches is estimated by surveying comparable problems in artificial intelligence. A truly artificial synthetic organic chemist that automatically constructs viable synthetic routes is clearly a challenging artificial intelligence problem and not directly amenable to existing approaches but chemistry could encourage new combinations of machine learning methods with automated reasoning to realize this goal. The importance of objective and open competitions with standardized problems and evaluations is also detailed as critical to realizing tangible computer programs that automate the planning of plausible synthetic routes.

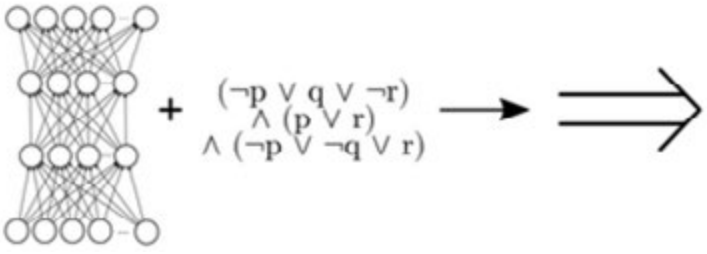


Figure 4. *Automated Scientific Reasoning.* Credit: Maneesh K. Yadav. The image below is the abstract of the paper "On the synthesis of machine learning and automated reasoning for an artificial synthetic organic chemist" by Maneesh K. Yadav, accessed Aug. 17, 2018[3].

[3] "On the synthesis of machine learning and automated reasoning for an artificial synthetic organic chemist" Maneesh K. Yadav. New J. Chem., 41, 1411-1416 (12 Jan 2017) 10.1039/C6NJ02492K

There is a lot of depth here in the AI, which I am not going to go into now, but I think it is a very important thing for a number of reasons. We see now increasing enthusiasm for AI in many different areas, and a lot of the energy and money is going into applying AI to problems that maybe in the grand scheme of things are not the most important problems. Historically most AI funding was military. Now most AI funding is pretty much going to try to get people to click on ads while they are surfing the web. It funds other interesting projects that the big tech companies are doing. On the other hand, there is huge potential to apply AI to designing new drugs, or designing new catalysts, new types of materials, that are arguably much more important for the future of humanity than putting ads on web pages. But due to the way the economy works, there is a lot less focus on these life-saving, world-changing applications of AI.

I am focusing on the AI algorithms part. I am doing all this OpenCog AI work and am dying to apply it to nanotech as I am to biotech and robotics. In order to get started on that, there is a lot of nitty-gritty work that must be done to make it easier. So what Christian is going to start with is talking about a *software platform* he has been developing, which is called CANDO, and is very elegant and allows integration of different simulation engines, different data processing software, and will allow integration of different AI software, although we haven't done that yet, into a common platform. So he will talk a bit about that, about his work applying that to *molecular design*, and if we still have time, I may induce Steve Fowkes come up and talk a bit about how you could apply the same type of toolset and software framework to a different type of nanotech, where you are doing atomically precise positioning, and building things with metal atoms instead of the larger molecules that Christian is working with.

Now Christian will take over and talk about some exciting work he has been doing that already realizes some of this vision, and which I think we can use to realize the rest of it.

Christian Schafmeister: Earlier today, John Koza was speaking about Genetic Programming (Fig. 5). He is the father of Genetic Programming; he wrote the book on it (Fig. 6); he wrote several books on it. One of the interesting examples he showed was how they used genetic programming to design circuits (Fig. 7).

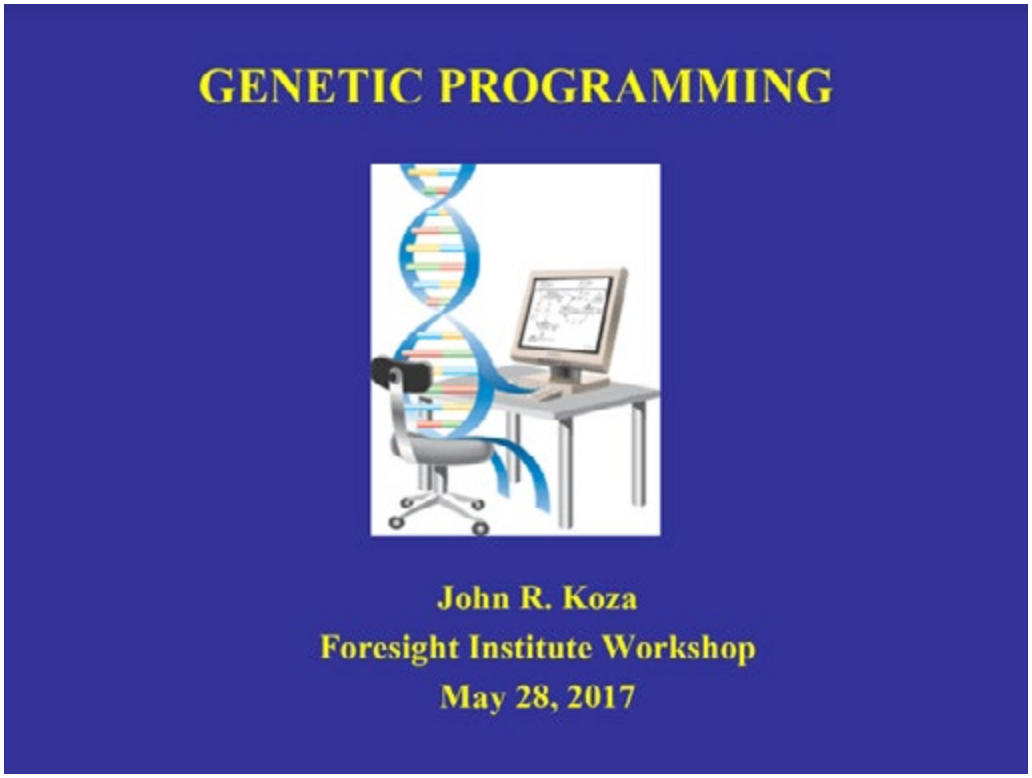


Figure 5. *Genetic Programming.* Credit: John R. Koza, used with permission.

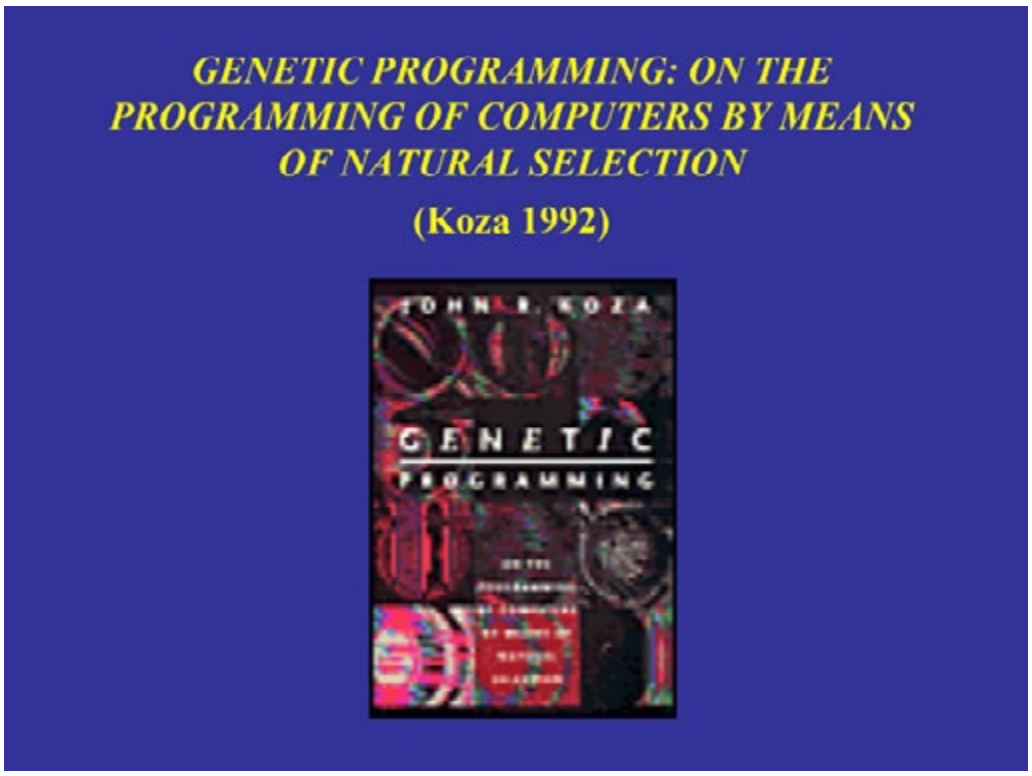


Figure 6. *Genetic Programming. On the Programming of computers by means of Natural Selection* (Koza 1992). Credit: John R. Koza, used with permission.

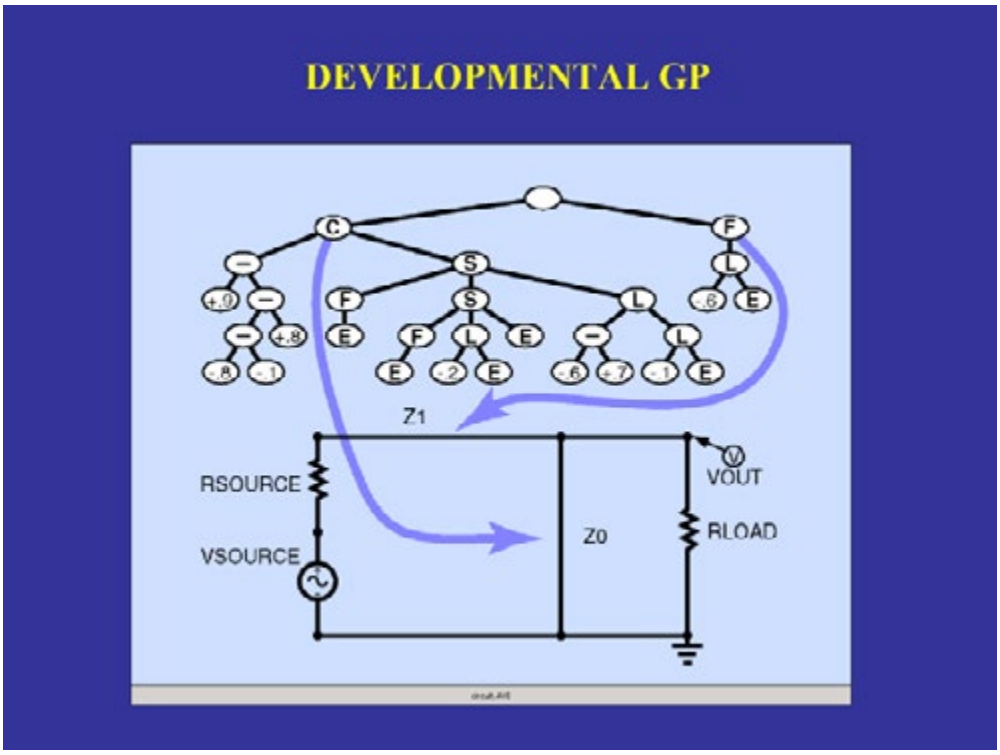


Figure 7. *Using Genetic Programming to design circuits.* Credit: John R. Koza, used with permission.

This is something that I have been interested in for a long time, and wanted to learn once I got to a point where my other systems were in place, but to use AI for molecular design. I am going to talk about two ideas (Fig. 8): (1) Software package to design molecules, (2) molecules that can be designed. Applications for these ideas include new medicines, catalysts, and atomically precise filters. I have a software package that can help everyone do molecular designs. I've also got a set of molecules that are very amenable to design. We can design functional molecules with them. The two of them together are going to be like a Reese's peanut butter cup. The applications that I am working on are to develop new medicines, new catalysts that can accelerate chemical reactions to make new products, and to do things like pull carbon dioxide and water out of the air and use sunlight to make them into hydrocarbons, and to make atomically precise filters.

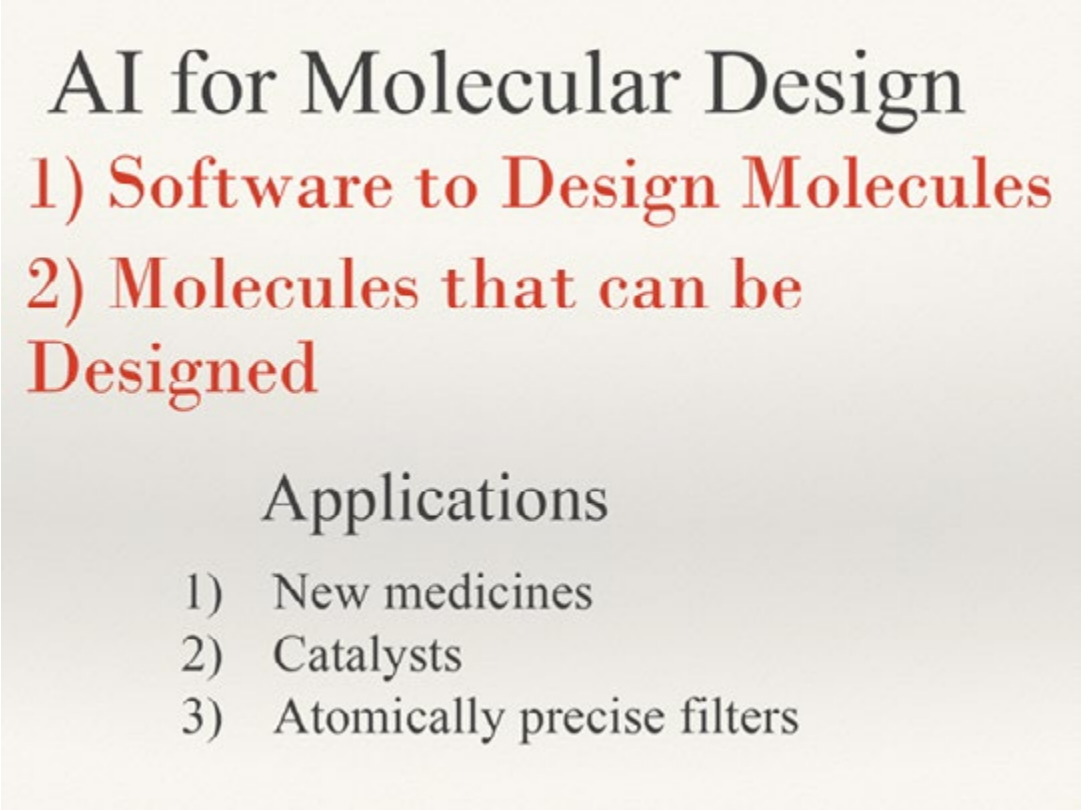


Figure 8. *AI for Molecular Design.* Credit: Christian Schafmeister, used with permission.

To use genetic programming to design molecules, you have to use evolutionary techniques, where you have a starting structure of a molecule, build a three-dimensional structure, score how well it can organize groups in space, then you mutate it, and depending on whether it scores better or worse, you keep it or you go back to your previous design, and you evolve your way to an optimal design.

To do this, you need to bring a lot of functionality together into one processor: you need the scoring function, you need building tools, you need a lot of software in one tight loop. In particular you need a molecular programming language where you can integrate all this stuff together to run in a tight loop to evolve your design.

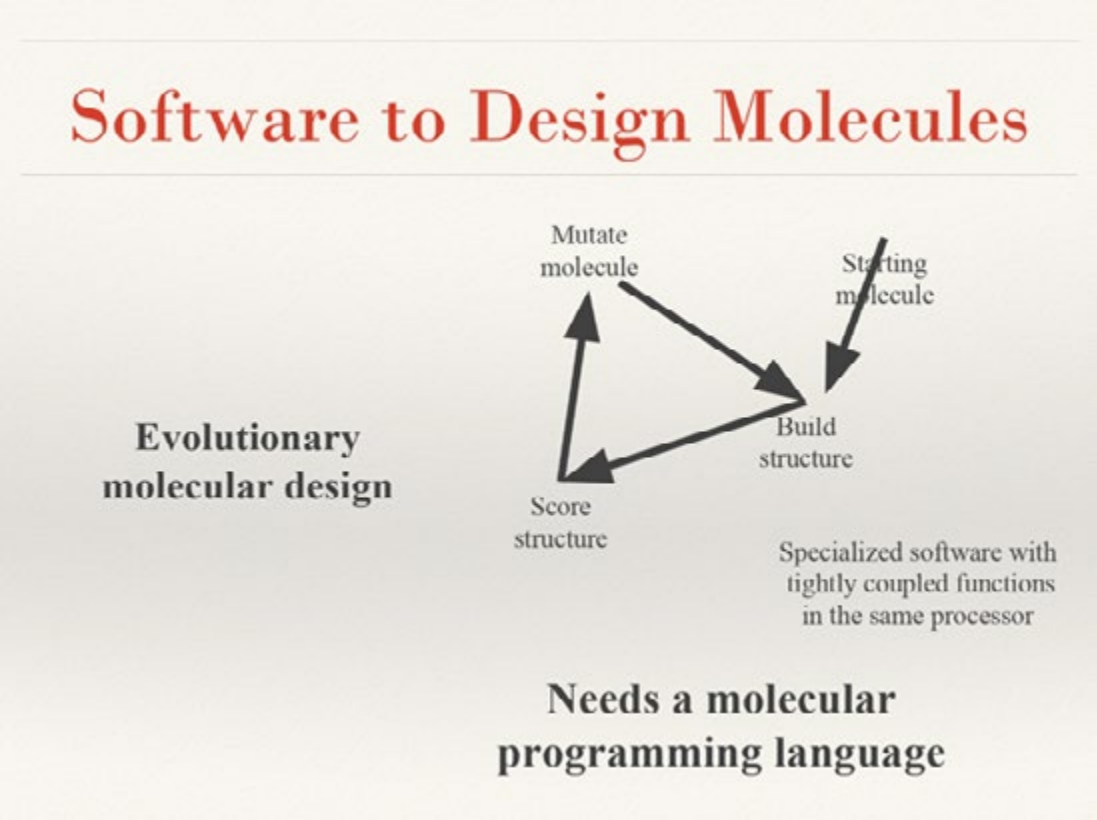


Figure 9. *Genetic Programming to Design Molecules.* Credit: Christian Schafmeister, used with permission.

I think the molecules that I have been developing to do this are much more amenable to design than are most other molecules out there. We call these things spiroligomers. They are building blocks that have a cyclic core, then two amino acids, one on one side and one on the other side. We can hang a functional group off of each building block that is like the side chain of an amino acid. When you link these to each other, they link through pairs of amide bonds. This is completely unique. There is no other chemistry out there like this. The result is complex three-dimensional structures that are kind of like sticks with functional groups hanging off of them, like protein amino acid side chains except that we can choose side chains from the entire chemical tool box.

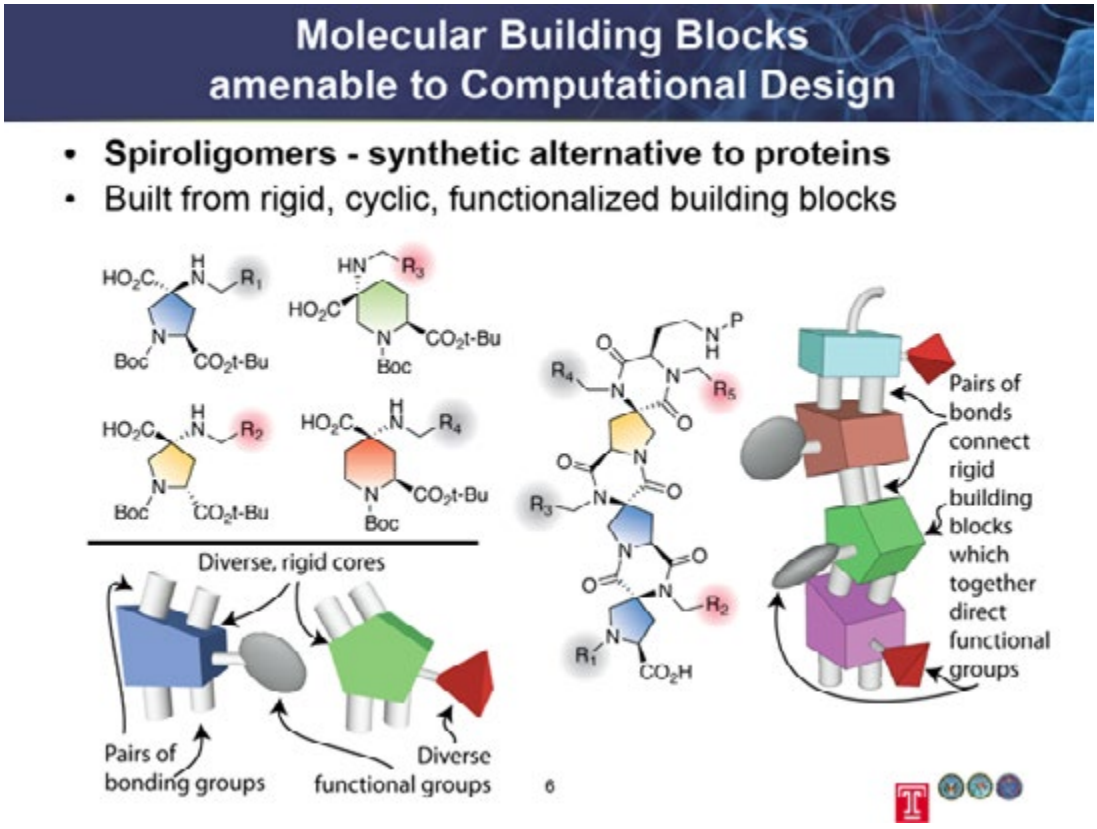


Figure 10. *Molecular Building Blocks amenable to Computational Design.* Credit: Christian Schafmeister, used with permission.

One of the things to which we want to apply evolutionary techniques is drug development. This is a cartoon of an HIV protein molecule (Fig. 11). If you could make a molecule that binds into this very shallow groove here, you could prevent HIV from fusing with and infecting human cells. A molecule that I would choose to use would look like this: a bunch of our building blocks strung together and connected through pairs of bonds. Everyone of these labelled carbon atoms here is a stereocenter. It is either in the S or the R configuration, pointing groups either into or out of the board. Every one of these side chains can be chosen from 100 different side chains, so there is an enormous design space here. You can imagine just randomly picking side chains and stereocenters, docking it against the protein, and pulling it away from the protein in a calculation to determine how tightly it binds. Then mutate a side chain and do it again. If you use evolutionary techniques like genetic programming you could have thousands of these molecules, run simulations in parallel, and then use sexual recombination to mix and match them, and then try it again, and try to optimize toward an optimal molecule. Then you make it and see if it binds to the protein.

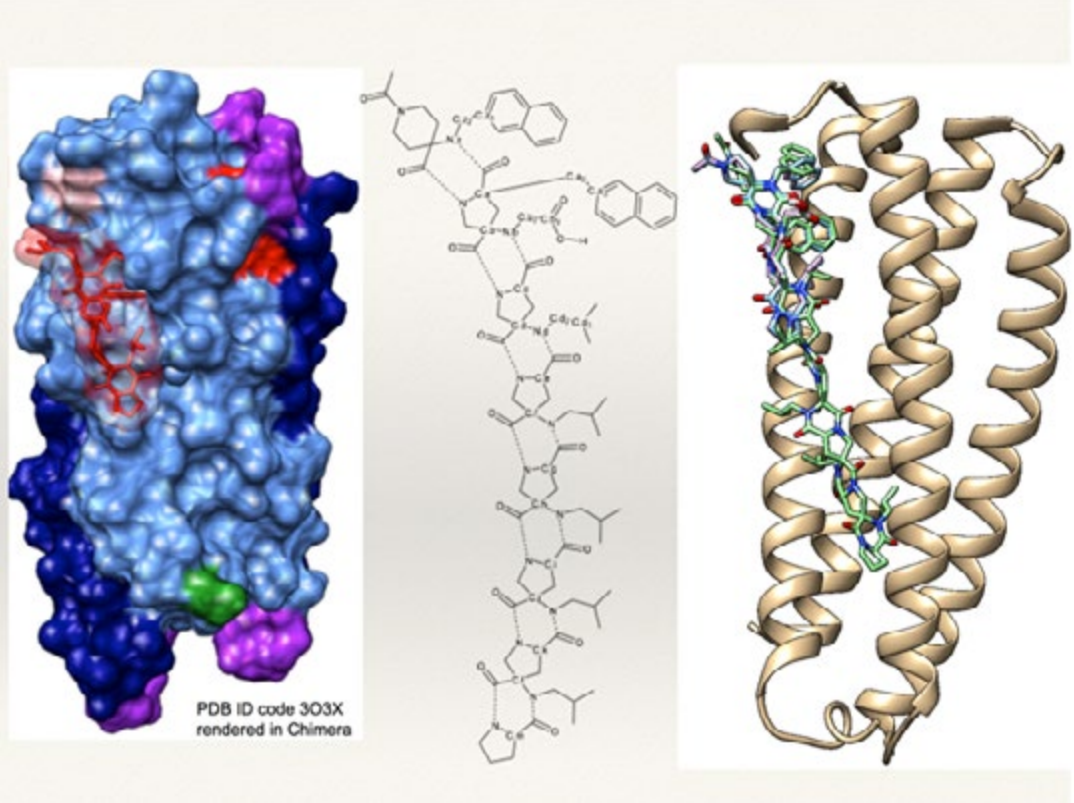


Figure 11. *Crystal structure of gp41-5, a single-chain 5-helix-bundle based on HIV gp41.* Protein Data Bank ID 3O3X. "Broad Distribution of Energetically Important Contacts Across an Extended Protein Interface" PubMed Central. Final edited form: J Am Chem Soc. 2011 Jul 6; 133(26): 10038–10041. Credit: Christian Schafmeister, used with permission.

The molecules have great drug properties: they are not going to be touched by proteases, they will not raise an immune response because they cannot be displayed on MHC proteins, and synthetically they are very straightforward to put together. We have been developing these molecules for about 15 years.

Another application is catalysts. Take the same kind of building blocks, different functional groups on the outside, synthesize these little 3-mers, and then stick them together like a peptide so it is four of these linked together. We crosslink them at the top using additional chemistry, and we present metal-binding groups off of different locations. This thing looks like a book or a clamshell. If you throw a metal in, and if everything is designed properly, then it could bind to the metal, have a pocket where substrates can enter and react only one way because they are controlled by the shell, like an enzyme active site. We can make trillions of these. We have a big design problem here: lots of stereocenters to control, lots of functional groups, so we need some sort of AI design tool because we cannot make them all. I think these molecules could be used to make enzyme-like catalysts that could be used, for instance, to change methane to methanol, which could change how we produce energy.

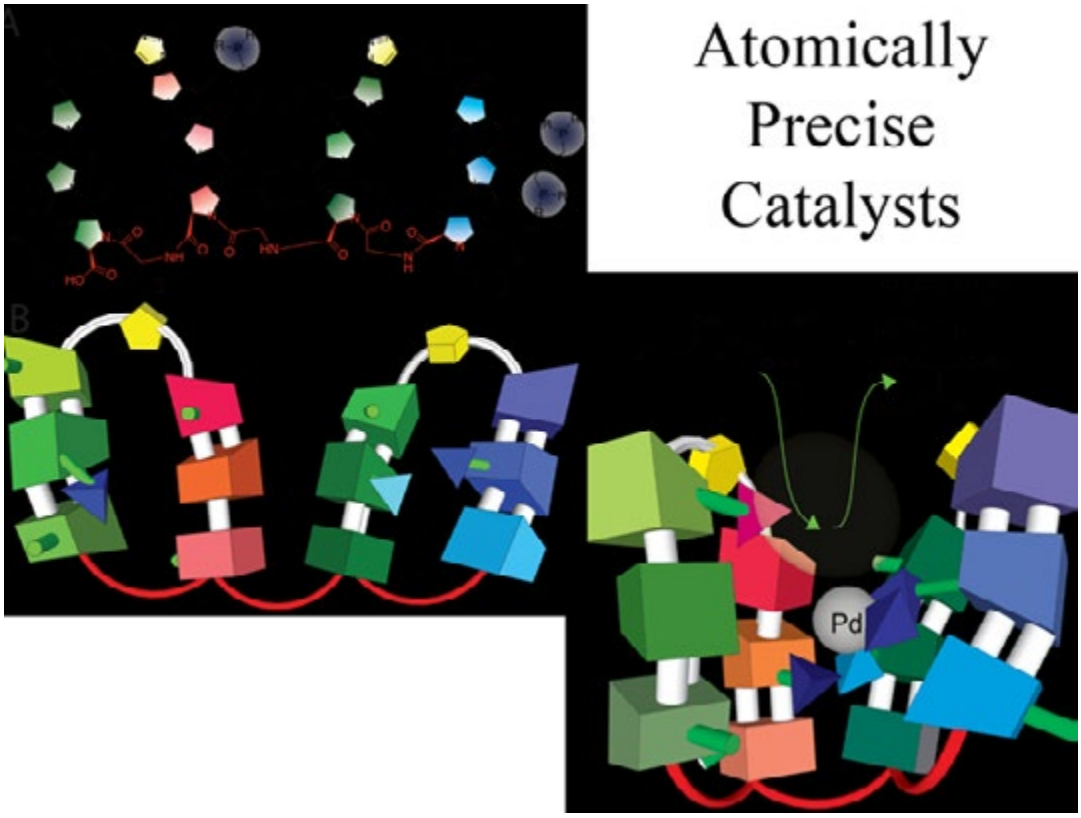


Figure 12. *Atomically precise catalyst fabricated from spiroligomers to bind a metal atom* (palladium, Pd). Credit: Christian Schafmeister, used with permission.

Another idea. This is something I am working on with the U.S. Army Corps of Engineers. They are very interested in trying to mimic aquaporins, the proteins in our cells that regulate osmotic balance (Fig. 13). We have already synthesized little triangles; what we are going to hang down are these long protheses, and then link them up with another triangle at the bottom to create a thing that is kind of like a triangular barrel (Fig. 14). With the software I developed I can build these molecules very quickly, build a two-dimensional network of them, and pop them into a box with water on the top and the bottom. Then the Aksimentiev group at the University of Illinois runs molecular dynamics simulations with NAMD (Nanoscale Molecular Dynamics) to calculate the flux through this membrane. It is about the thickness of a cell membrane. Then we can mutate the internal structure and see how it changes flux. Then we can computationally design channels for specific purposes, for water or organic molecules. I would love to be able to make big channels to separate small molecules. I would love to make a filter that can separate my building blocks from each other, and then have a box so I can just do the chemistry, run the product into a box with a membrane, and have four pipes coming out with each of the four purified building blocks. This simultaneous purification of multiple molecules is achievable with membranes like this.

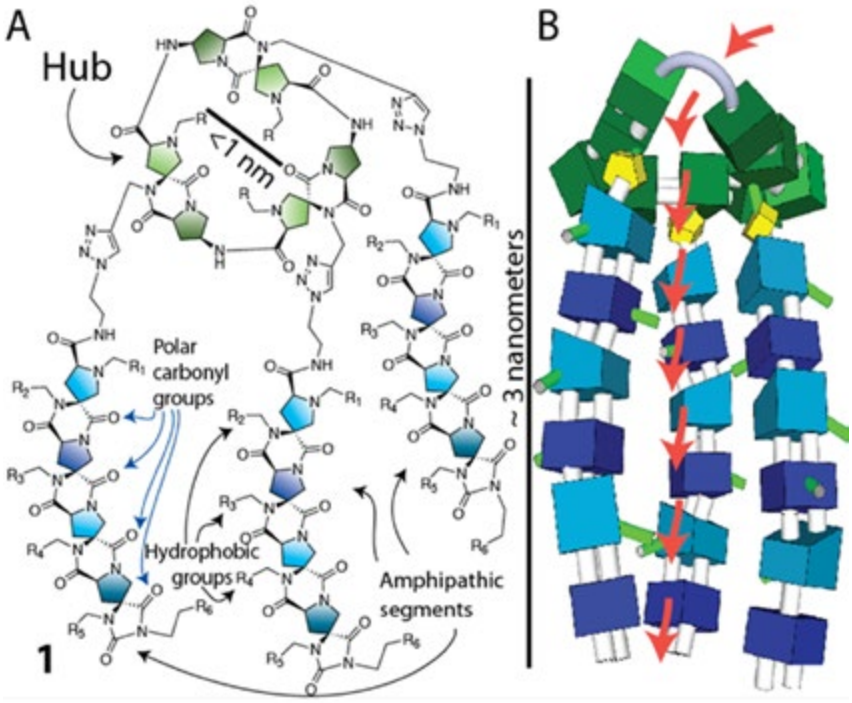


Figure 13. *Spiroligomers to mimic aquaporins:* integral membrane proteins that form pores in the membrane of biological cells to facilitate transport of water between cells. Credit: Christian Schafmeister, used with permission.

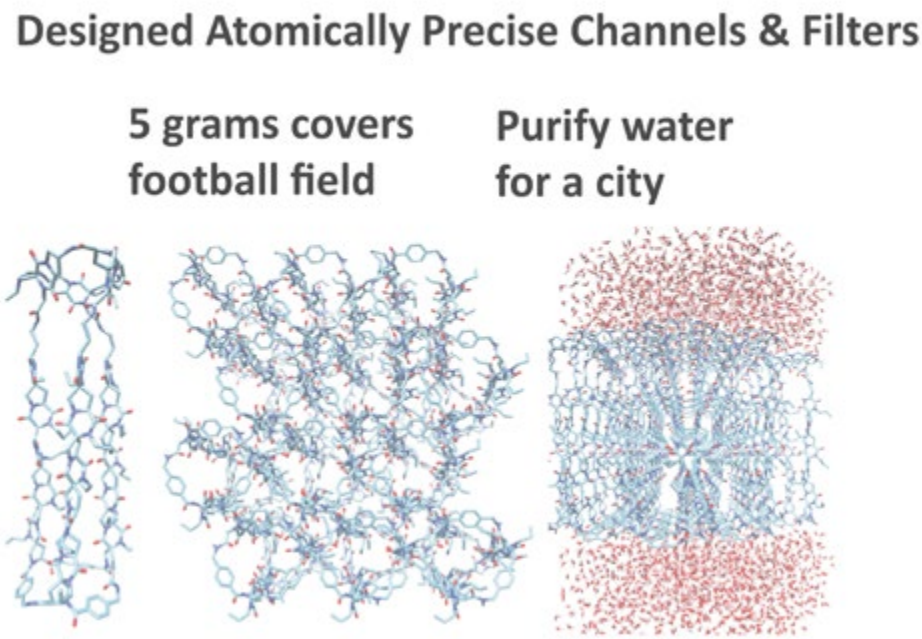


Figure 14. *Atomically Precise channels and filters made from spiroligomers.* The filters would be about 3 nm thick, approximately the thickness of a mammalian cell membrane. Credit: Christian Schafmeister, used with permission.

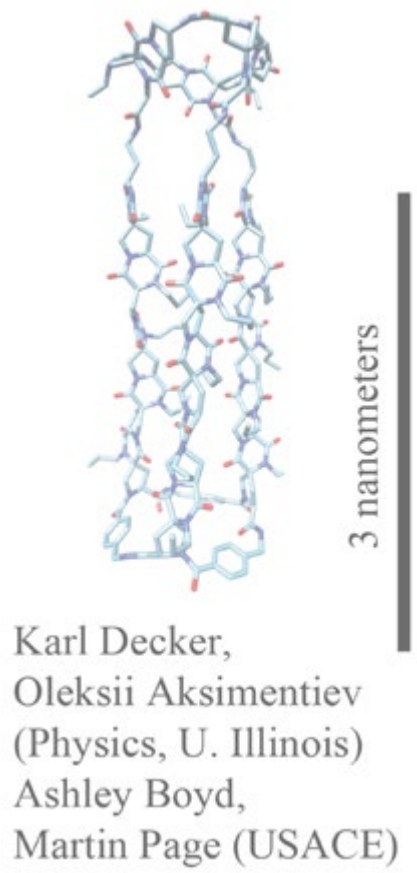


Figure 15. *Molecular dynamics simulation of water flux through a filter made from spiroligomers.* The filters would be about 3 nm thick, approximately the thickness of a mammalian cell membrane. Credit: Christian Schafmeister and the Aksimentiev group at the University of Illinois, used with permission. The animation that occupies the right-hand portion of Fig. 15 can be seen in the YouTube video NanoMind: AI for nanoscale design - Project proposal between approximately 18:44 and 19:34, and in a related video presented by Prof. Schafmeister at the European Lisp Symposium, 9-10 May 2016: CANDO: A Compiled Programming Language for Computer-Aided Nanomaterial Design and Optimization between approximately 17:00 and 21:30.

What this needs is new software. There are lots of software engines that are basically C++ libraries that do molecular dynamics or DFT calculations. I do not want to have rewrite those things. I want to use them. But the way you have to use them right now is you have to basically set up a file, run the program, load the file, run the calculation, it writes out a file, and then you have to write another program to analyze the results. It is very slow. It is not amenable to the tight building loop that I want to do.

So what I have been working on, and we have been talking about, is how to integrate these programs into a single unified executable that could have LAMMPS inside of it, and tools to build molecules, and then do this molecular design using LAMMPS or using AMBER, or using any of these molecular simulation programs (Fig. 16). What you want is something where you can write a program to describe the functional requirements for the molecule, run it through the package, and you will get a sequence of the molecule that, when you synthesize it, will have that function.

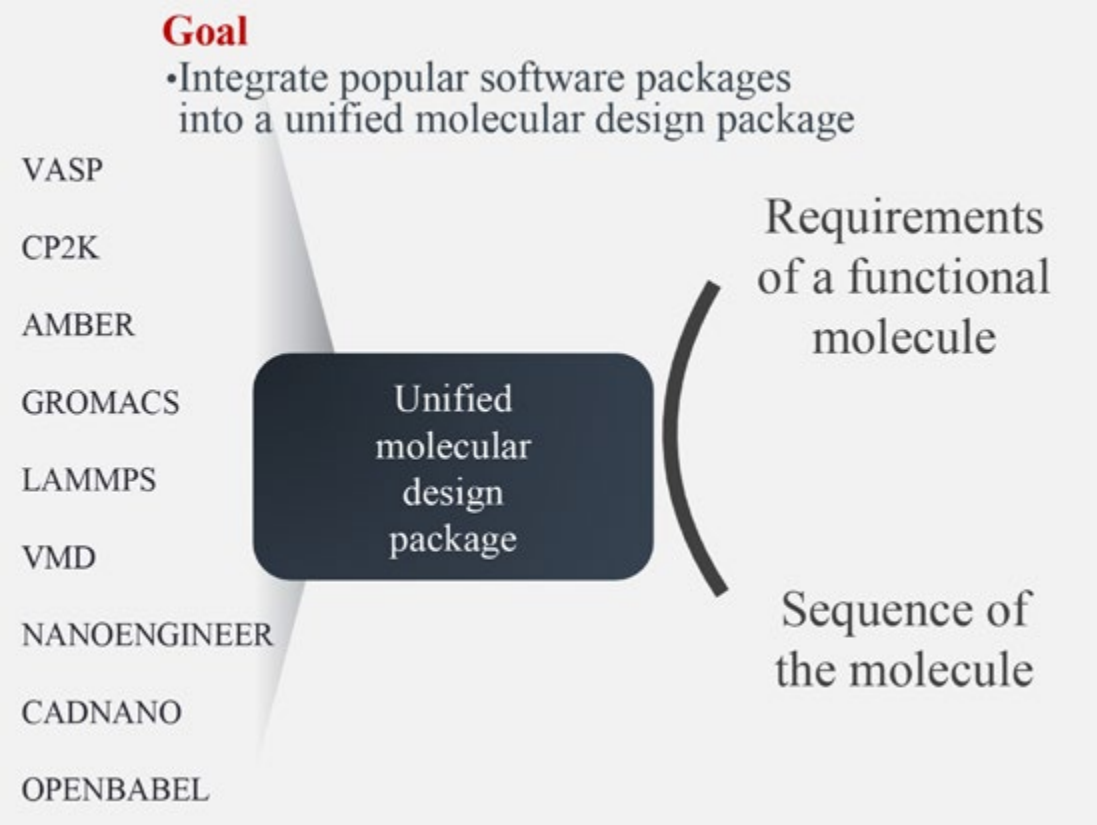


Figure 16. *A unified molecular design package.* Credit: Christian Schafmeister, used with permission.

I have been developing a software package like this; it is called CANDO (Fig. 17). I stands for Computer Aided Nanostructure Design and Optimization. It is a programming language. It is basically a whole bunch of chemical functionalities that manages atoms, molecules, residues, stereochemistry, force fields, automatic type assignment, and automatic charge assignment. It is all written in C++, but it has within it a Common LISP compiler that I have written myself (Clasp) that generates fast, native code. It is all based on C++ and LLVM; the library that makes iPhones and everything on a Mac is compiled with the LLVM library. CANDO and Clasp generate LLVM code, and that allows this (Fig. 18): you can basically take C++, or C, or FORTRAN code, compile that code into this intermediate representation that LLVM uses (LLVM-IR code). Common LISP is also compiled to the same intermediate representation. So you can link code together from C++, common LISP, and FORTRAN, and have it all inlined within each other to generate code that runs as fast as possible, with barriers as low as possible. This is the big, central engine behind CANDO. LLVM is being developed by Apple, by Google, by Sony Playstation because it is a very powerful compiler writing library, and it is built into CANDO.

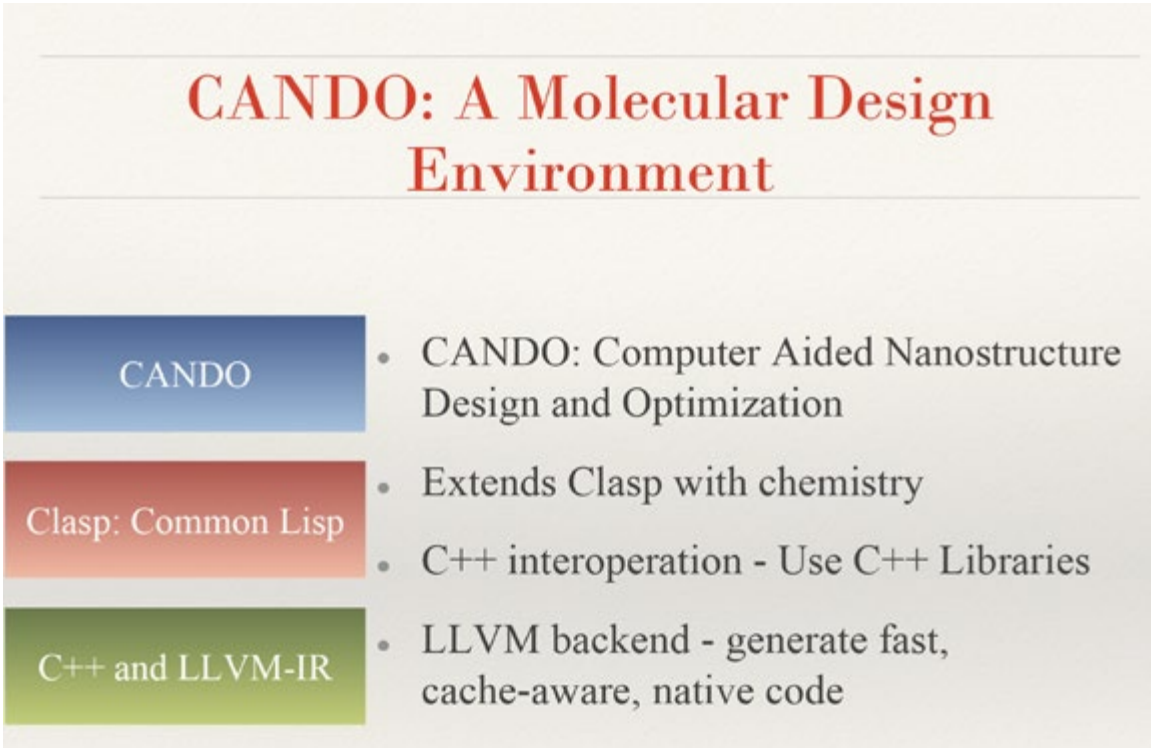


Figure 17. *CANDO: A Molecular Design Environment.* Credit: Christian Schafmeister, used with permission.

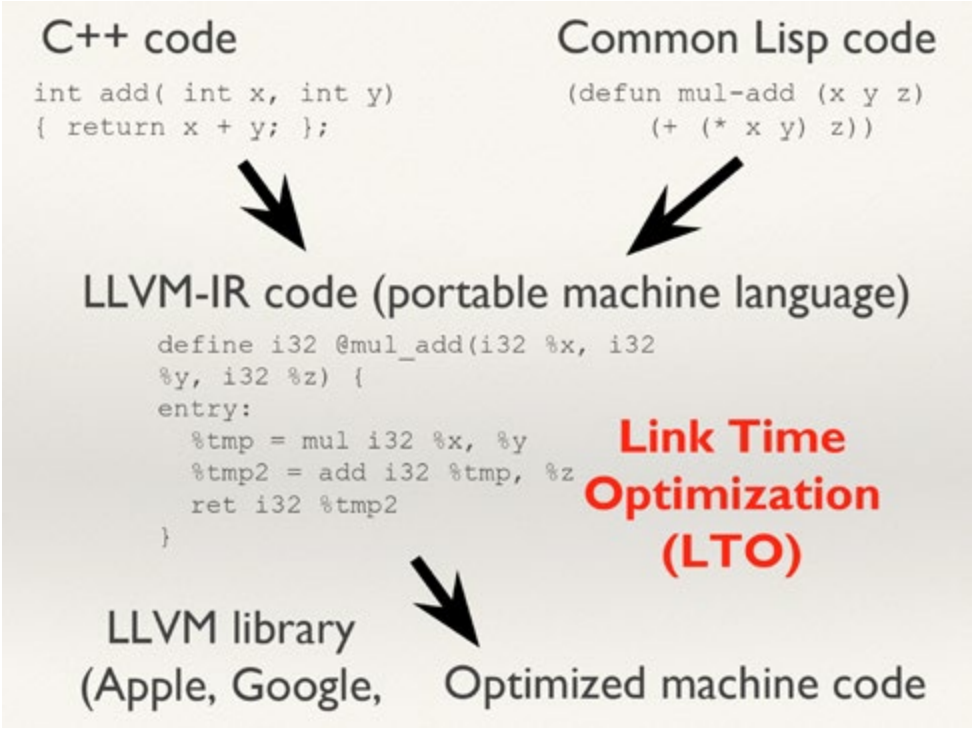


Figure 18. *From portable machine language to optimized machine code.* Credit: Christian Schafmeister, used with permission.

We are going to be releasing CANDO in the next couple of months. Basically, what it lets you do is go into ChemDraw and sketch a molecule you want to design (Fig. 19). This is a cyclic molecule that has our little building blocks on it, and I want to hang a group off of here, and here, and here, and all point to the same side because what I want this molecule to do, and I am funded by the Department of Defense to do this, is to coordinate a zirconium oxo metal cluster that can hydrolyze nerve agents. They want catalysts to hydrolyze nerve agents. So we want something that can coordinate a metal cluster to protect it so it can act like a soluble enzyme. So we need a molecule that points these three groups on the same side at the right distances from each other. The things we can control are we can change the stereochemistry here, here, here, here, here, here. We can change what group we hang off there, there, there, here, here, and here. We have a lot of variables that we can mutate.

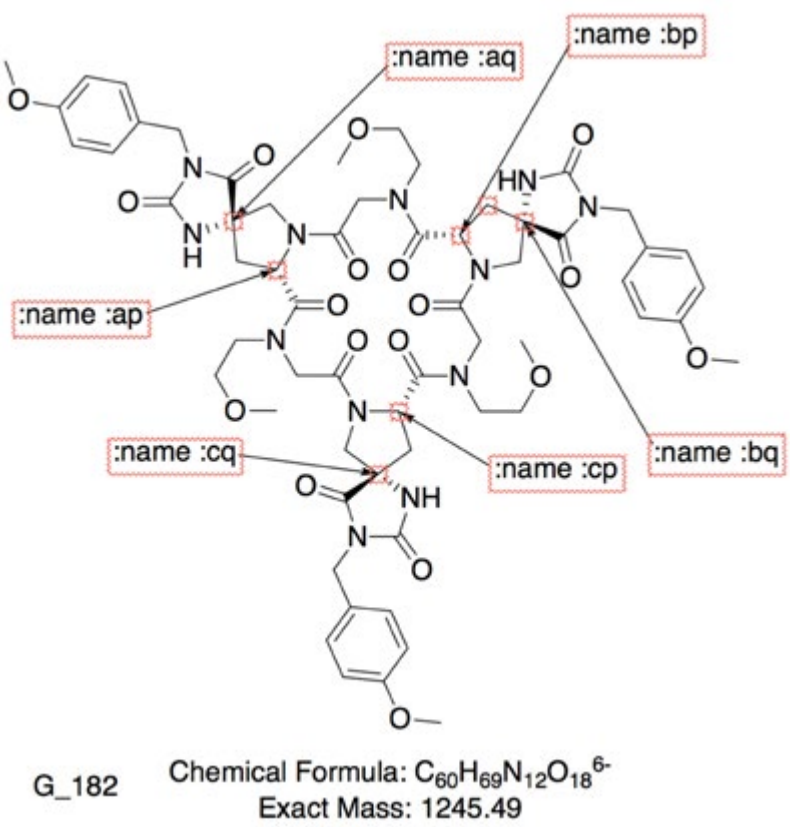


Figure 19. *Spiroligomer-based molecule to coordinate a metal oxo cluster to hydrolyze nerve agents.* Credit: Christian Schafmeister, used with permission.

The way I am setting this up (Fig. 20), I can load that ChemDraw file into CANDO; that's what is running here. Your interface is what is called a Jupyter Notebook. You can write a code in CANDO that will build every stereoisomer, do energy minimization on it, calculate whether the carboxylic acids that are generated match up to where they would need to be to bind the zirconium oxide cluster, then iterate through them all, and save the best fit, giving the result of the calculation.

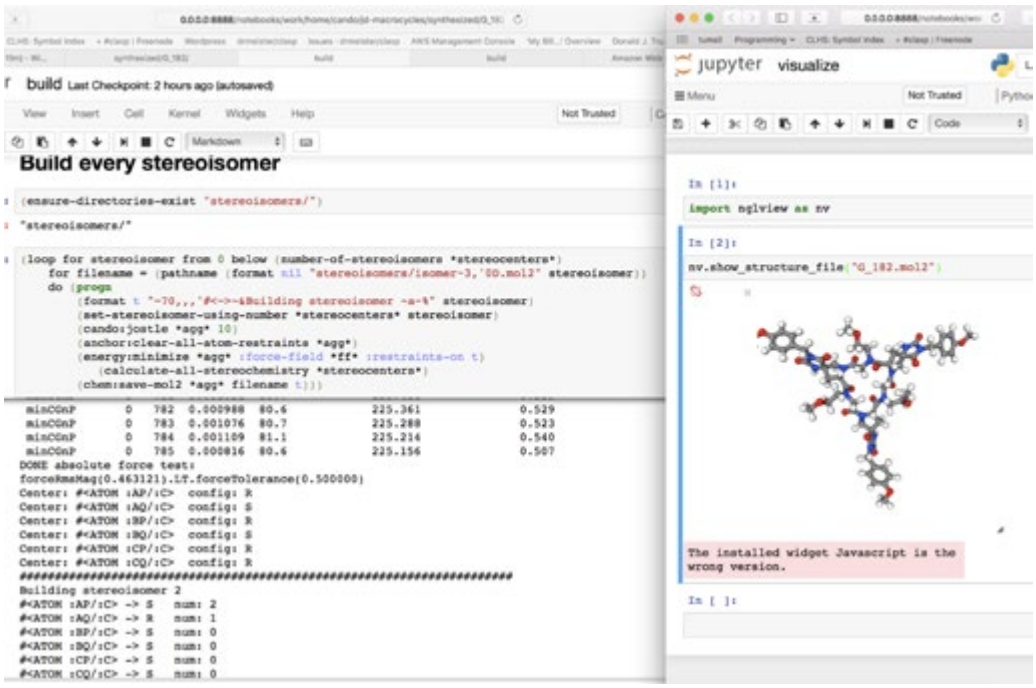


Figure 20. *Evaluating stereoisomers in a Jupyter notebook.* Credit: Christian Schafmeister, used with permission.

What we are doing now is getting the widget stuff to work inside the Jupyter Notebook so you can sketch the molecule in here, run the calculation, and see the structure down below. It will provide a very nice user interface for setting up and sharing molecular calculations. You can design all sorts of molecules with it, anything that is made out of atoms it can help you build, model, and design.

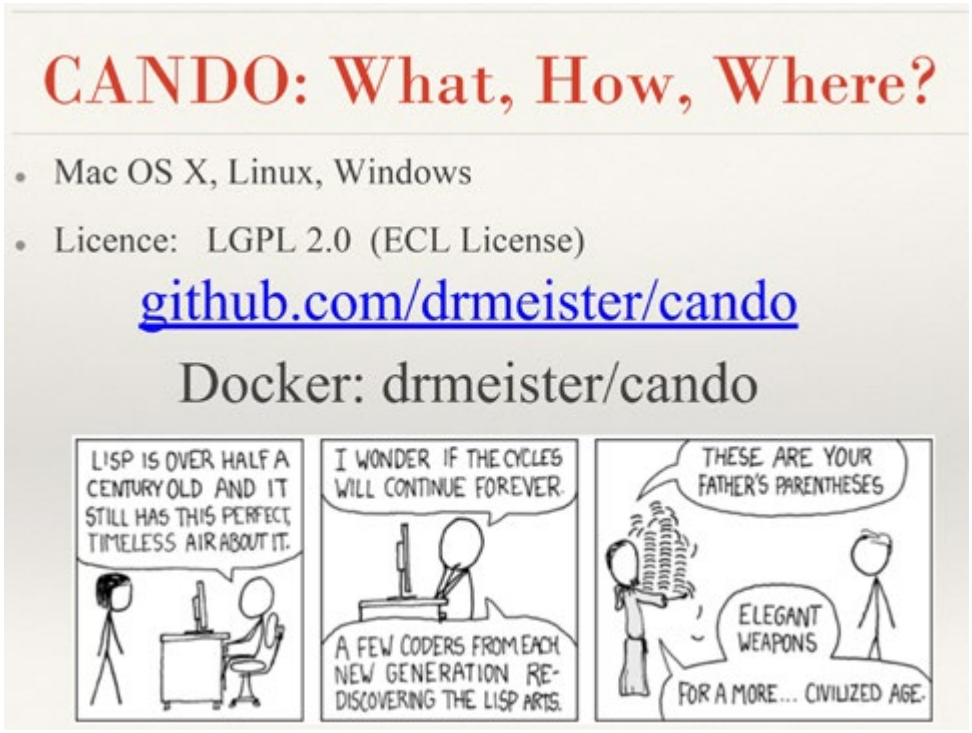


Figure 21.
CANDO: What, How, Where? Credit: Christian Schafmeister, used with permission.

Next steps, as suggested during discussions after the presentation: Start experimenting with short-term applications before tackling longer-term, more ambitious goals. See below for two useful examples in each space:

1) Short-term. Filter: Filters which would let certain substances through and not other substances through could be designed via genetic programming, and deep learning. One could start designing simple filters that would not let CO2 through, or more complex filter combining benzene and cyclohexane rings with some extra complexity to allow self-assembly so that urea would not be able to pass, but water and certain other things could. Those filters would be directly valuable to the developing world and the developed world, but from an AI point of view it is interesting because it is an easy example, where one can specify the substance that is allowed to pass, the one that is not allowed to pass, and the AI system explores the kinds of structures with molecular dynamics simulations to model how well each structures served complies with those parameters, e.g., how much of the desired substance it lets through and how little of the undesired substance.

2) Longer term: Enzyme Cavities: Engineered enzymes in cavities using the special building blocks that Chris Schafmeister has produced.

For the detailed discussion following the presentation please go here in Part 4: Bonus Material.

Part 2: Background information

As illustrated in Part 1, AI for Nanoscale Design is an extension of Christian Schafmeister's CANDO software to incorporate genetic programming and OpenCog's framework, as well as incorporating a greater variety of simulations, platforms and other external AI tools. The goal is to create a unified software environment which eliminates the need to learn and use different tools manually, but allows easy and unified use of different simulators, and different AI algorithms that serve different functions, such as design or accelerate a simulator. This section provides background information on the building blocks for AI for Nanoscale Design:

1) An advanced software framework for AI-powered nanoscale design, which is proposed in the early draft of the NanoMind proposal

2) the CANDO framework, an existing software tool for chemistry, which is proposed as technical foundation for NanoMind

3) Genetic Programming, an existing machine learning technique, which is proposed as a potential AI tool to expand on existing software solutions to chemistry.

1) NanoMind: A Software Framework for AI-Powered Nano-Design

Ben Goertzel
Hanson Robotics, Aidyia Limited, AGI Society and OpenCog Foundation

Below is the very rough draft of NanoMind, as submitted by Ben Goertzel to Foresight Institute in March 2017. It is based on combining the proposal Machine Learning Based Atomistic Simulation for Nanostructures and Processes with the proposal Machine Learning to Accelerate Quantum Chemistry Simulations from Foresight's workshop on AI for Scientific Progress.

Introduction and Vision

The creation of powerful nanotechnology presents numerous challenges. The hardware and wetware challenges have historically been foremost among these, and are still in the process of being overcome. However, there are also serious challenges in the area of system design. Human intuition has not been tuned by evolution for nanoscale system design ("nanodesign"), meaning that the nanodesigns humans conceive may well be profoundly far from optimal. Porting macroscale designs to the nanoscale and imitating molecular biology with slight variations are bound to be highly limiting methodologies.

The most promising route to working around these limitations is to apply artificial intelligence to the nanodesign problem. AIs do not have intuitions overfitted to the everyday macroscopic world; they can think about the nanodesign problem with open minds and can be expected to come up with solutions that would elude the human intuition. To a human being, quantum phenomena are bizarre and unnatural and hard to incorporate into design thinking; to an AI, the regularities of the quantum world are simply mathematical patterns to be observed, manipulated, and mastered.

We do not yet know what AI techniques will be most powerful for nanodesign, nor what classes of nanodesigns will be most amenable to AI-based design techniques. What is needed most at this stage is a general, flexible, efficient and simple-to-use platform for experimenting with various AI techniques in the context of nanodesign for various classes of nanosystems. Furthermore, this platform should be open source and free for all so that it can unrestrictedly be improved and adapted by the scientific community.

This is the motivation for the NanoMind system proposed here. NanoMind, as envisioned, will be a software framework for connecting AI systems, nanosystem design specifications and constraints, and nanosystem simulators such as quantum chemistry simulators. Such a framework will make it far more straightforward than is currently the case for AI researchers to try out their software on nanodesign problems, and for nanotech researchers to try out AI systems to solve their design problems. Bringing the two fields and communities together in this way can be expected to lead to a variety of novel and unexpected discoveries.

NanoMind Architecture Overview

The key components of any specific utilization of the NanoMind architecture, as proposed here, will be:

1) Simulator: a quantum chemistry simulator

2) Constraints: a set of specific constraints on a class of nanodesigns (e.g., only carbon atoms, only DNA molecules, etc.)

3) Specification: a specification of the requirements for a desired nanosystem and a function specifying to what fuzzy degree a given nanosystem satisfies the requirements

4) AI Algorithm: an AI algorithm, which will iteratively search for designs satisfying the constraints and specifications, using the simulator (because the "function" mentioned in Step 3 will normally need to use the simulator as a sub-function)

5) Visualizer: a visualization tool (bearing in mind that the optimal visualization tool could be different for different types of nanosystems)

6) Evaluator: The NanoMind core enables loading of constraints and specifications, and evaluation of proposed designs according to constraints and specifications (using the simulator)

7) Control Interface. The capabilities of the other components may be accessed via software commands or manually via a web user interface. For instance, two of the many required functions will be:

a. If an AI algorithm is registered with the control interface, the interface can then be used to request the AI algorithm to propose a series of designs based on particular constraints and specifications, which will then be evaluated by the Evaluator component (and the evaluation results will be supplied to the AI algorithm and also stored by the control interface and shown to the user upon request)

b. If a design is proposed by an AI algorithm, or uploaded by a human user, the control interface can then be used to request a visualization by the Visualizer component

The initial version of NanoMind will then consist of:

- An architecture for connecting components of these 7 types
- Some initial specific components fitting into each of these 7 slots, providing useful examples of the architecture's functionality.

We propose to implement NanoMind as a set of modular components interacting via two custom languages:

A. NanoComm: a communication language (using an existing lower-level communication protocol) for exchanging information among the 5 different components. This should be human-readable but will be generated by software, not authored directly by humans. XML is one possibility but multiple alternatives will be evaluated.

B. NanoScript: a scripting language for specifying items 2 and 3 (constraints and requirements). This may be implemented as a small DSL (domain-specific language) inside an existing programming language, perhaps Python or Scheme. Scripts in this language may be authored directly by humans, or output by software.

We will then create proxies enabling existing software to communicate via NanoComm and NanoScript.

For the Simulator component, there are numerous options, but currently BigDFT[4] seems the most appealing with its capability to leverage both GPUs and distributed processing.

For the AI Algorithm component, it will be valuable to compare multiple AI approaches on a common nano-design problem-set. We propose to test the following three approaches:

- Genetic Programming, which has been applied successfully to numerous problems in circuit design, factory design, etc.[5]

[4] http://bigdft.org

[5] See John Koza's book Genetic Programming III: Darwinian Innovation and Problem Solving for an in-depth discussion of GP-based circuit design; and David Goldberg's book The Design of Innovation for discussion of numerous other design applications of genetic algorithms and evolutionary computing

- The OpenCog[6] integrated cognitive architecture, which incorporates a genetic-programming-like tool (MOSES), along with other components such as a probabilistic reasoning engine.
- A deep neural network, likely a variant of stacked InfoGAN[7] (chosen because of its relatively transparent, disentangled representation, which gives a way to inspect the internals of the network and see what it is doing)

Some comments on the AI learning strategy proposed, for use with all three of these AI approaches, will be given in the following section.

The Visualizer will need to be created in a custom way using existing open-source visualization components. It will be desirable to create a visualization component that can be viewed in a Web browser, for ease of use and platform-independence.

Finally, regarding the Constraints and Specification, which naturally work together, we consider it important to consider at least two test problems: one involving "wet" nanotech based on organic molecules, and one involving "dry" nanotech based on atomically-precise positioning of atoms and smaller molecules.

AI for Nano-Design: Principles, Resources and Tools

The NanoMind framework is designed to be agnostic regarding the particulars of the AI algorithms utilized for learning. Any AI algorithm that can propose designs in the NanoScript language and communicate them to the Evaluator component using the NanoComm language, will be able to carry out learning within NanoMind.

That said, however, AI learning of nanodesigns is a challenging problem and certain general approaches are bound to be more successful than others, apart from the algorithmic specifics. A few principles that we suspect will be valuable in this domain are:

1. To use AI to learn procedures for building nanosystems, rather than explicitly learning completed designs
2. To evaluate the quality of a candidate procedure both in terms of properties of the procedure and properties of the completed design
3. To estimate the quality of a nanodesign before feeding it into a quantum chemistry simulator or other expensive simulation algorithm, via probabilistic reasoning about this nanodesign based on data gathered via evaluating other nanodesigns
4. To simplify procedures for building nanosystems, or simplify completed nanodesigns, based on formalized knowledge of (or approximations to) the laws of physics in the relevant context
5. To generate training data for AI algorithms via

[6] See http://opencog.org, and the books Engineering General Intelligence Vol. 1 and Vol. 2 by Goertzel, Pennachin and Geisweiller

[7] InfoGAN is described in the paper InfoGAN: Interpretable Representation learning by Information Maximizing Generative Adversarial Networks by Xi Chen et al (https://arxiv.org/abs/1606.03657, Submitted on 12 Jun 2016); new variants of stacked InfoGAN are emerging rapidly and it's not clear which one will be most successful (we are developing one within the OpenCog project also).These networks have been designed and refined for computer vision and will need significant modification for application in a nanotech context.

a. Creating models of natural systems, importing them into NanoMind, formulating the constraints and specifications they follow, and asking AI algorithms to learn these natural systems (or viable alternatives) based on the constraints and specifications
b. Formulating constraints and specifications of interest, then using crude brute-force search to find nanosystems fulfilling these constraints and specifications to various degree. The results of the brute-force search can then be used as training data, the goal being to have AI algorithms that can find these solutions much faster than brute force search did.

These ideas provide suggestions for tools that should be useful for diverse AI algorithms operating within the NanoMind framework. For instance, virtually all AI algorithms used in this context would benefit from a training dataset based on natural systems and brute-force search, providing examples of the form.

((optional) procedure, design, constraints, specification, quality)

Most AI algorithms would benefit from a Fitness Estimator component that used probabilistic inference (or nearest neighbor matching, or any other method) to guess the quality of a design before evaluating it with a full-scale simulation. Such a component could be used to prevent expensive evaluation of candidate designs that are "obviously" unlikely to be viable based on prior knowledge.

A Procedure Simplification component would also be useful to any AI algorithm that operated by learning procedures for constructing nano-designs. Different AI algorithms might represent procedures in different ways, but given a common set of assumptions regarding the construction machinery to be used, there is bound to be significant commonality. A powerful tool for simplifying construction procedures based on a simple formalism for combining the relevant mechanical primitives would almost surely be broadly useful.

The nature of fitness estimation, procedure simplification, and training dataset construction are bound to be significantly different for diverse types of nanosystem (e.g., wet versus dry). However, the mode of utilization of these tools within NanoMind can nevertheless be the same regardless of the particulars of the problem being addressed.

Example of an Envisioned Nano-AI Workflow: OpenCog for Nanodesign

To make these concepts more concrete, we will now review how these might be used in an OpenCog context. This section in its current form may be somewhat opaque to readers not familiar with OpenCog and its components, but unfortunately, this is somewhat unavoidable given the complexity of the topic and processes involved.

Similarly detailed decompositions may be made for the application of other AI approaches to nanodesign. The NanoMind framework is designed to be as agnostic as possible regarding AI approach and nanotechnology approach; however, the OpenCog approach is the one we have thought through most thoroughly at this stage.

One would use OpenCog's MOSES (genetic-programming-like) component to learn small computer programs for controlling nanomachinery to construct a nanosystem. To do this, MOSES would need to be supplied with primitives corresponding to the movements and perceptions of the nanoscale construction machinery in question.

MOSES's search-based expansion of program trees may be augmented by a more planning-like approach, in

which PLN[8] inference is used to create high-level plans for creating nanosystems fulfilling the given specifications, and probabilistic adherence to such a plan is taken as part of the fitness function for MOSES.

Procedures created by MOSES would then be fed to the Procedure Simplifier to see if they could be reduced into simpler form. The Simplifier might also be called during MOSES's internal learning process, to guide learning.

The Procedure Simplifier itself (or one version thereof) could be implemented using OpenCog's internal rule engine, with the physical rules of appropriate types of nanosystem implemented as ImplicationLinks (rules) within OpenCog, and PLN (Probabilistic Logic Networks) inference used to make uncertain inferences regarding possible simplifications.

PLN's guesses regarding possible simplifications could be used to guide crisp physical inference regarding simplifications; this is a special case of a general workflow via which PLN may be applied to guide theorem-proving.

Candidate procedures/designs conceived by MOSES would have their quality estimated by the Fitness Estimator component to assess their viability. Designs passing this stage would get sent to the Evaluator component for evaluation by a quantum chemistry simulator or similar.

PLN would be used as one option within the Fitness Estimator, to carry out fitness estimation based on information regarding previously evaluated procedures and designs, leveraging the OpenCog Pattern Miner and potentially also MOSES within its processing.

The major factor controlling the speed of PLN inference is "inference control"—choice of which inference steps to take—and the best approach to this within OpenCog is history-based. Inferences are stored and pattern-mined, and patterns characterizing successful inferences are re-used to guide future inference. Some of these patterns are generic but many are domain-specific, and learning patterns of successful PLN inference regarding nanodesign is something that will occur only after applying PLN to a number of nanodesign problems (including a large number of simulation problems as proposed above).

2) CANDO Framework

Christian Schafmeister
Department of Chemistry, Temple University

To create NanoMind, Christian Schafmeister's CANDO software would be extended to incorporate genetic programming and OpenCog's framework, as well as incorporating a greater variety of simulations, platforms and other external AI tools. CANDO provides a Lisp-based unified framework that wraps together different simulations and other software, which is in C++, Fortran and other languages. Creating a unified software environment is the central technical goal of CANDO.

Please find the transcript of Christian Schafmeister's workshop presentation on CANDO at Foresight's AI for

[8] Probabilistic logic network Probabilistic Logic Networks: A Comprehensive Conceptual, Mathematical and Computational Framework for Uncertain Inference by Ben Goertzel, Matthew Iklé, Izabela Lyon Freire Goertzel, Ari Heljakka (2008) Springer. p. 333. ISBN 0-387-76871-8.

Atomic Precision workshop below. His slides can be found here.

A related video presented by Prof. Schafmeister at the European Lisp Symposium, 9-10 May 2016: CANDO: A Compiled Programming Language for Computer-Aided Nanomaterial Design and Optimization

I want to discuss our work and the motivation for it. Spiroligomers (Fig. 22) are synthetic alternatives to proteins that are built from rigid, cyclic, functionalized building blocks. These are cyclic amino acid monomers that have an amino acid on one side, an amino acid on the other side, and you can hang a functional group off of them like an amino acid side chain. These are like Lego groups that connect to each other through pairs of amide bonds so that the kinds of molecules that we end up making are ladder molecules. Functional groups stick out of them at weird directions based upon the stereochemistry of the center, the shape of the rings, and the order in which we put them together. We can make all sorts of different presentation of functional groups off of the sides of the molecule. This is the core technology my group has been developing the past 15 years.

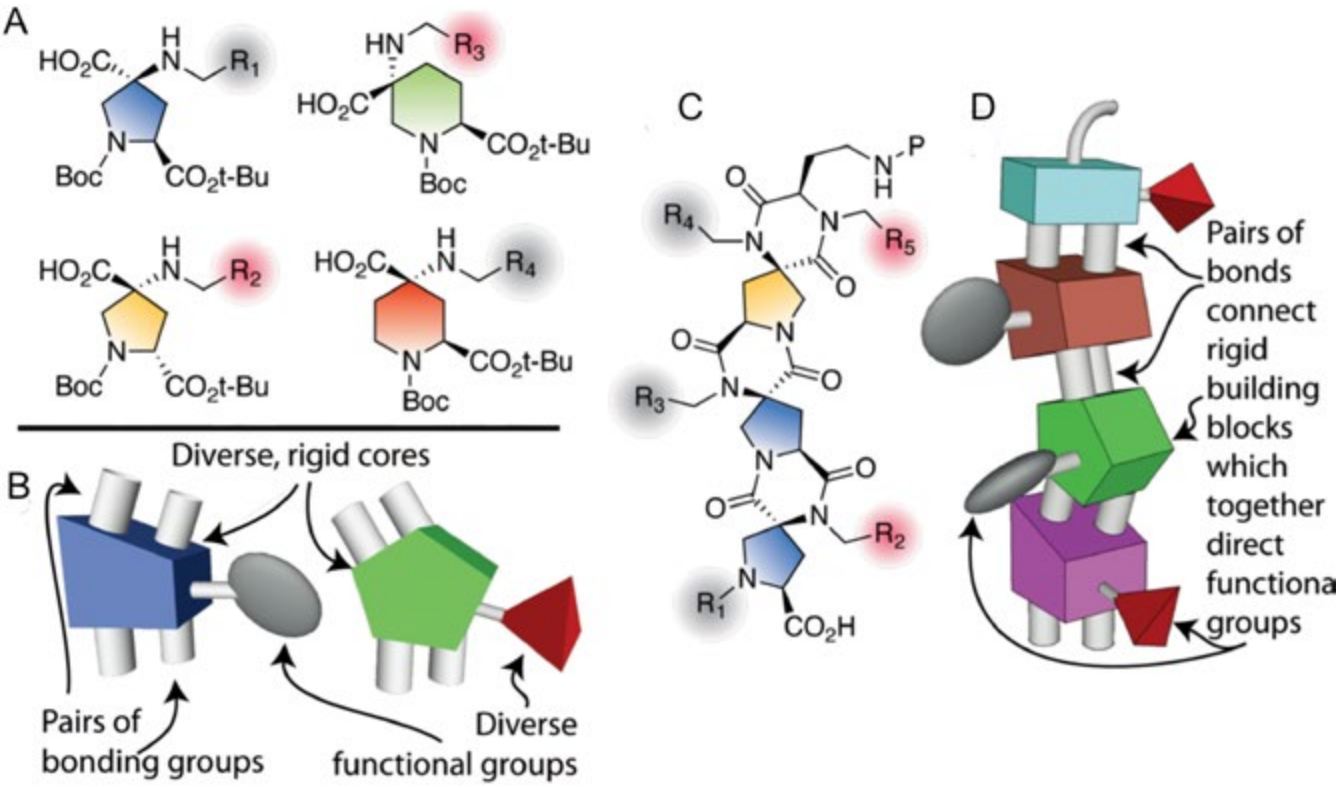


Figure 22. Spiroligomer Scaffolds. "(A) bis-amino acids (B) 3D Mock up of a bis-amino acids (C) Spiroligomer Trimer (D) 3D Mock up of a Spiroligomer."
https://commons.wikimedia.org/wiki/File:Spiroligomer_Scaffolds.png
This file is made available under the Creative Commons CC0 1.0 Universal Public Domain Dedication. Author: WingmanJ

Some of the molecules we have made are catalysts[9,10]. This is a molecule that has three of our building blocks (Fig. 23). They are all set up with the right stereochemistry to present a base, an alcohol, and a urea group in the proper configuration to do a transesterification reaction. You convert one kind of an ester into another one. The ester comes in here, this base tugs on this proton, making it more reactive, it attacks this carbon here, generates a negative charge here that is stabilized by the hydrogen bonding group. This molecule accelerates the first step of the reaction about 2500 times, and it does multiple turnovers. It is like a little catalyst.

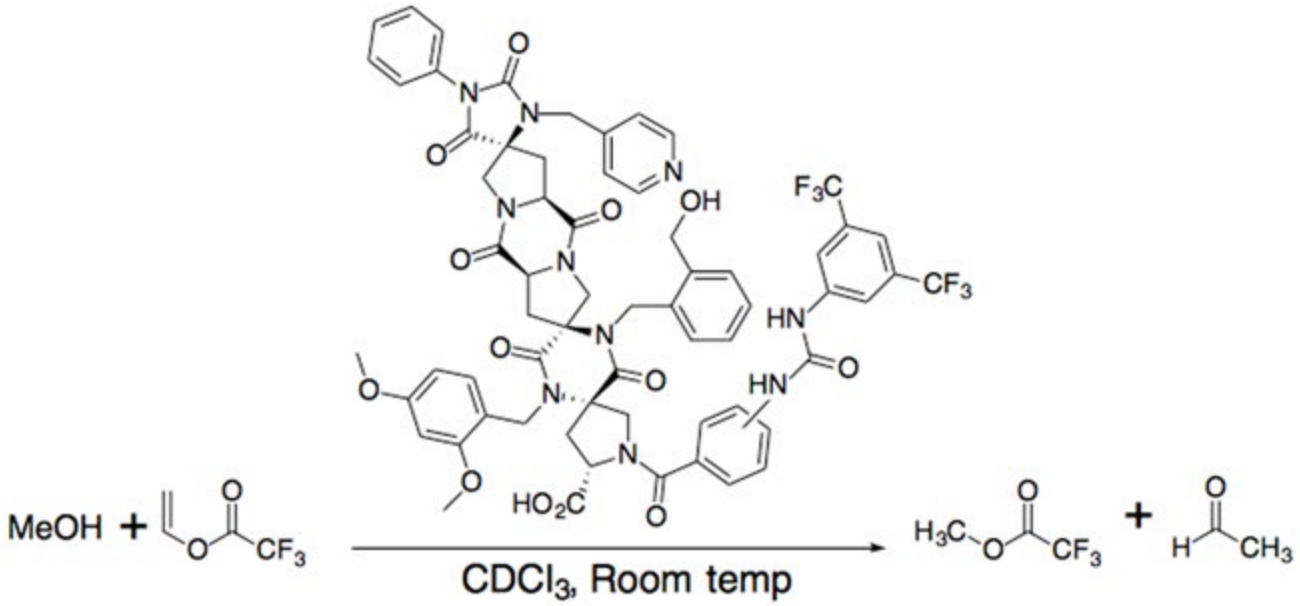


Figure 23. Transesterification catalyst. Transesterification of vinyl-trifluoroacetate with a Spiroligozyme
https://commons.wikimedia.org/wiki/File:Transesterification_Catalyst.png
This file is made available under the Creative Commons CC0 1.0 Universal Public Domain Dedication. Author: WingmanJ

This catalyst was rationally designed using the earlier version of the software we have designed, CANDO. We have designed another molecule with just the right stereochemistry to be half of a better antibiotic. But what we really want to do is to take multiple building blocks and tie them together into bundles to create molecules with pockets inside of them. If we can present metal atoms in the pockets to create metal catalysts that are surrounded with a shell to control what gets in, how it orients, and how it reacts inside of the pocket, then we can create atomically precise catalysts that can do things like convert methane to methanol, which would be huge.

This is our plan: we can synthesize these three building block segments, we can string them together like peptides, crosslink them at the top, present metal binding groups at particular positions, arrange the stereochemistry properly,

9. "Spiroligozymes for Transesterifications: Design and Relationship of Structure to Activity" M Kheirabadi, N Çelebi-Ölçüm, MFL Parker, Q Zhao, G Kiss, KN Houk, CE Schafmeister. J. Am. Chem. Soc. **134**, 18345–18353 (September 19, 2012) DOI: 10.1021/ja3069648

10. "Acceleration of an Aromatic Claisen Rearrangement via a Designed Spiroligozyme Catalyst that Mimics the Ketosteroid Isomerase Catalytic Dyad" MFL Parker, S Osuna, G Bollot, S Vaddypally, MJ Zdilla, KN Houk, CE Schafmeister. J. Am. Chem. Soc. **136**, 3817–3827 (January 23, 2014) DOI: 10.1021/ja409214c

bury them in something like a clamshell, and throw a metal atom in to pull the binding groups together, and create a pocket so that it creates one product and not the other, because the shape of the pocket can control the chemistry.

This is what is happening inside our bodies right now. Human beings cannot make things of this size, this complexity right now. We need to make things like this and explore them. This is the motivation behind everything that I do. But the problem is that these are enormous molecules. We have 20 stereocenters in here. That is 2 to the 20th power possible stereoisomers. Every position here we can put any one of a hundred functional groups. The number of possibilities is astronomical. Any one of them we can synthesize in a couple of weeks, but which one should we make?

So I need an oracle. I need something to specify what the requirements are for the functional molecule, like an iron atom in the bottom of the pocket with molecular oxygen, then something around it to control what hydrocarbons can get at it, and how they orient to make an oxygen transfer. Then we can take a very cheap alkane pumped out of the ground and turn it into alcohol worth a thousand times what the alkane is worth. But we need a tool to go from the requirements for the molecule to the sequence of the molecule.

How do computers design molecules? One way is to build a lot of candidate molecules and then run them through a series of successive filters in order to eliminate some molecules and then pass them on. That works with small molecules and small libraries, but if you have big molecules like the ones we will be making, you have to go to something like Monte Carlo simulated annealing, where you start with a random starting point, and then you successively mutate and score until you reach a local optimum. You can set this up on thousands of processors to search through the design space to find molecules that have a particular function.

However, the software to do this doesn't really exist. There is a lot of molecular mechanics and molecular simulation software out there, but something that can couple billions of the kind of molecules that we are synthesizing, or that other people are synthesizing, with running a little simulation on them, scoring them, then mutating them, and doing that in a very tight loop, that did not exist.

So I am writing it, and it is called CANDO, and what it is, is basically an implementation of Common Lisp that has half a million lines of C++ code that does everything an organic chemist knows about molecules. The programming environment that implements Common Lisp has a lot of chemistry code built into it. It is implemented in C++ but it uses LLVM as the underlying library. CANDO is a compiler that generates fast layer code, fast machine code to do everything. You can take any C++ library, like for a neural network, for molecular dynamics, that other people have written, and very quickly expose it inside of this language, and then write programs that use the libraries, but they run very fast because they are also compiled in machine language.

This is the kind of thing I want to do. This (see Fig. 19 above) is the prototype for a molecule we wanted to design to bind a zirconium-oxo cluster named MU1000. It is being developed to hydrolyze nerve agents, to make metal organic frameworks. We would like to make a water-soluble version of this thing. The idea here is to make a ring that has four groups on it that will point three carboxyls to one side in the right position so they can coordinate this metal complex. So I go into ChemDraw and I sketch this molecule as a prototype. I have stereo centers that I can flip to either S or R, which changes the shape of the molecule. But I do not want to build all of these combinations by hand; I want to build them programmatically. What CANDO does is load my ChemDraw structure, built a 3D structure for it, and then I can loop through, build every stereoisomer and do a geometry optimization in a tight loop. In a couple minutes it will generate all 2 to the 64 structures that I can run dynamics simulations on, score each one, and pick the one that organizes the groups in the best way.

The fundamental technology underneath all of this is LLVM. It is a compiler writing library. I figured the only way we could do this is to implement the program language where we could use existing C++ libraries to do simulations of molecules, and hook them in with a high level garbage-collected language that you could use to drive that stuff, but that generates native code that links together. I see this as a system of software components that are linked together. It's a programming environment; it's a compiler; it's a lot of chemistry code; it also has tools that can link other software into it and run it inside of the environment where you can pass molecular structures from one function to another, in memory, so you don't lose time and efficiency.

For more information:

- The CANDO computational chemistry environment
- CANDO and CLASP Overview
- You Tube video presentation by Christian Schafmeister CANDO: A Compiled Programming Language for Computer-Aided Nanomaterial Design and Optimization

3) Genetic Programming

John Koza
Genetic Programming, Inc

NanoMind is an extension of the CANDO software to incorporate genetic programming and OpenCog's framework, as well as incorporating a greater variety of simulations, platforms and other external AI tools. Rather than just building a software architecture that makes it easy to experiment with different simulators and with different algorithms in different places, genetic programming would allow one to go a step further: One could use evolutionary learning combined with probabilistic inference to generalize the evolved models, a refinement of the idea of using genetic programing, or use genetic algorithms to learn to structure deep neural networks.

Please find the transcript of the presentation on Introduction to Genetic Programming, given by John R. Koza at Foresight's AI for Atomic Precision workshop below. The complete set of presentation slides is available here.

A video of a similar lecture on genetic programming given by John R. Koza at Stanford University in 2013 can be found here: Automated Design Using Darwinian Evolution and Genetic Programming.

I will say a few things about genetic programming that might be relevant to things you are doing. Basically genetic programming is a way to replace an antiquated thought-based method with evolution. What genetic programming does is to generate computer programs, because computer programs are the common language of solving all sorts of problems. The main point that I would like to convey to you is that genetic programming now routinely delivers high-return human-competitive results, and notably in inventing things and designing things, which I think is the part that you are most interested in. And it does so with a minimum of human input, so it has what we call a high A to I ratio, that is, the ratio of what is produced automatically (A), divided by the amount of intelligence (I) the human user put in. It has produced a series of results that have been in synchrony with the Moore's Law increase in computer power.

When we talk about 'human-competitive' we mean many things, but for this group, it is producing better than a human can design. One example of that is to produce patented inventions. That was the thing that we latched onto to prove that the technique could produce routinely human-competitive results. We went back to the patent literature of different fields and reinvented things based only on the specs of what the patent was meant to produce. There has been a succession of results that has pretty much been in lockstep with Moore's Law, starting with trivial toy problems from the late 1980s, some human-competitive non-patent results, some 20th century patented results, some 21st century patented results, and then some things that produced new patents.

In the first book (*Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Koza 1992) we talked about evolving programs. Programs are things that have inputs and outputs. They may have subroutines, loops, recursions, and storage of different types. We used the analogy of genetics (Fig. 24) applied to computer programs, so we randomly generated a large population of random computer programs, we see how good they are at doing whatever you want them to do, and then we select individuals from that population to slightly mutate some programs at random, but more importantly, to take pairs of programs, and cross over pieces of one program to the other. To randomly create programs we use Darwinian selection. The main points from the 1992 book are shown in Fig. 25.

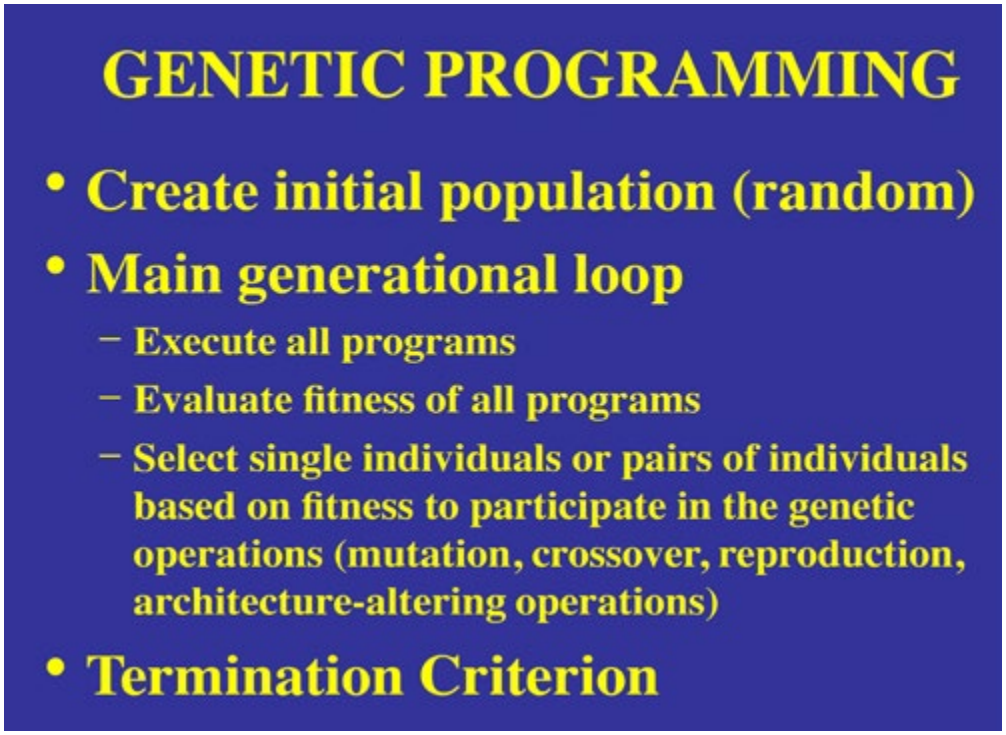


Figure 24. *Genetic Programming.* Credit: John R. Koza, used with permission.

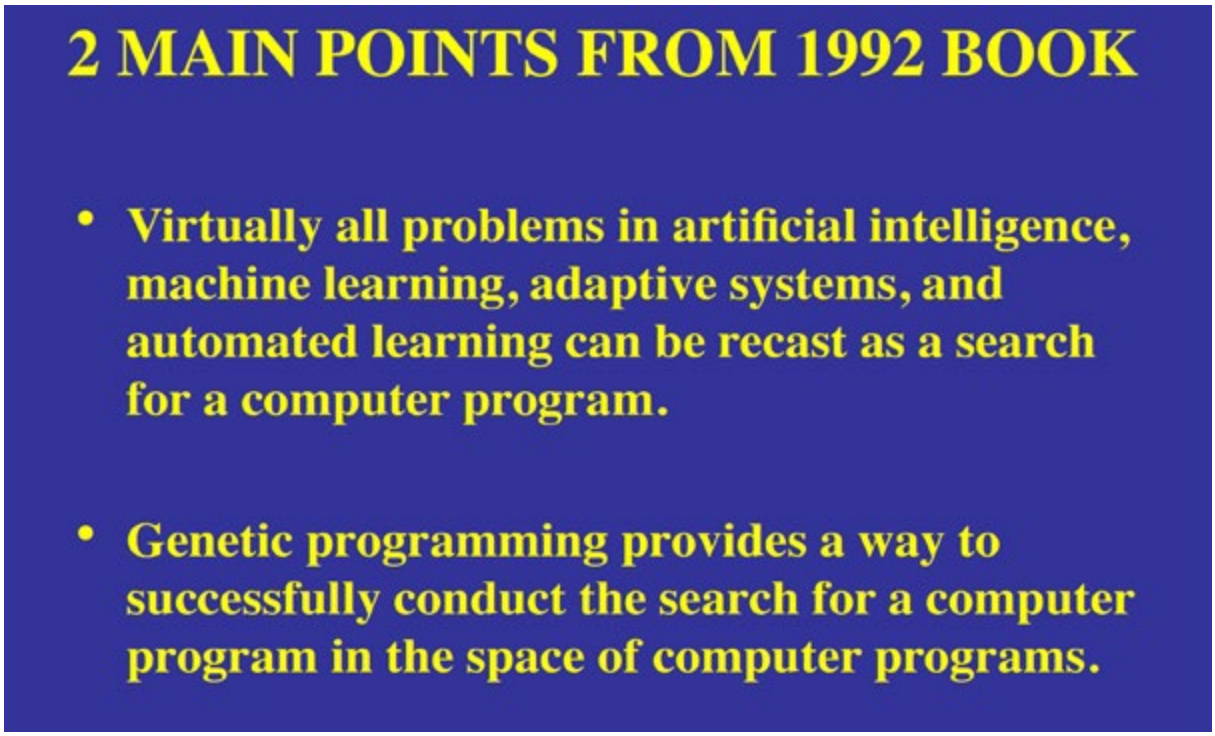


Figure 25. *Two Main Points from the toy problems discussed in Book I, Koza, 1992.* Credit: John R. Koza, used with permission.

In the second book (*Genetic Programming II: Automatic Discovery of Reusable Programs,* Koza 1994) the key concept was reuse, that is, subroutines. If you want to compute the difference in volume of two boxes, there is a common calculation: multiply length by width by height of the two boxes. You will learn quicker, and you certainly will code quicker, if you take that subroutine to calculate volume and reuse it twice instead of inventing that wheel twice (Fig. 26). The book contains all kinds of demonstrations as to how these automatically defined functions, or subroutines, allow you to reuse code and to scale things up desirably.

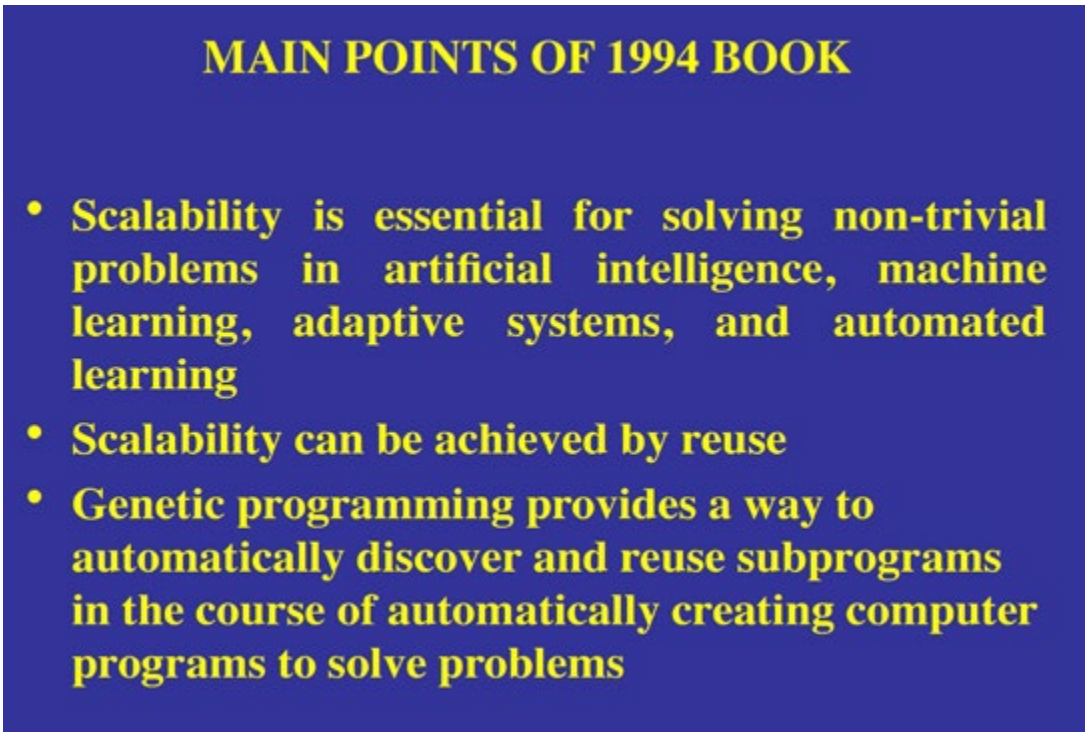


Figure 26. *Main Points of 1994 Book.* Credit: John R. Koza, used with permission.

In 1999 we got into the business of replicating patented inventions (*Genetic Programming III: Darwinian Invention and Problem Solving*, Koza, Bennett, Andre, Keane 1999). This work includes stuff that I think is really relevant to the kinds of things that you are trying to do here. We had a number of examples of replicating early 20th century, late 20th century, and early 21st century analog electrical circuits. By the way, there is no other automated way to create designs for these circuits. We used a developmental process. So this is not just evolving programs. We are not talking about evolving computer programs that contain instructions for constructing something. This is a developmental process loosely following the natural process. For example, a wire in a circuit can be turned into a component with a function in a computer program that says "convert a wire into a capacitor", or some other circuit component. We can also modify the topology of that wire by turning it into two parallel wires. We can take an embryonic circuit that does nothing and by a series of component-inserting and topology-modifying functions, we are converting it into something else. We have a program tree. We have an embryonic circuit, which does nothing. We expand the program tree to create the fully designed circuit. We put that into the simulator to see how well the circuit performs, Then we put that fitness into the genetic algorithm. The result in this case is the Campbell filter that was patented by Bell Labs in 1917.

In the 20 years after 1917 there were dozens of filters named after dozens of other people, that had special additional characteristics, and we were able to recreate them with those desired characteristics. In the 1950s, transistors came along. So what is the change we had to make to go from re-inventing the greatest hits of the early 20th century electrical engineering world vs. the second half of the 20th century? We threw out the inductor and we inserted the transistor. That was the only change. We started evolving transistor filters, including negative feedback, one of the seminal inventions of control theory and electrical engineering, in a completely automated way, just going from the requirements of the problem. Then we went through the literature of 21st century electrical engineering

patents and came up with a bunch of others.

Another technique we arrived at was subroutine duplication. How did nature get hemoglobin from myoglobin? There was a divergence at some point, and somehow a very similar chain to myoglobin got put together four times, and we got hemoglobin, which had a different function, binding oxygen in the bloodstream instead of binding oxygen in the muscle. Using that analogy, we came up with subroutine duplication, where we take a program, we take a duplicate of a subroutine. In evolution if you have two copies of a gene, one of which is essential, you can fiddle with the other through a process of mutation, crossover, and other processes, and evolve a new functionality, as happened with hemoglobin and myoglobin.

Part 3: Introduction to Artificial Intelligence & Molecular Machines

To complement the proposal AI Nanoscale Design (part 1), and the background information on the proposal (part 2), this section focuses on an introduction to AI and Molecular Machines for those interested in a primer on the scientific fields addressed by NanoMind.

Introduction to Artificial Intelligence

Ben Goertzel
Hanson Robotics, Aidyia Limited, AGI Society and OpenCog Foundation

Please find the transcript of Ben Goertzel's introduction at the AI for Atomic Precision Workshop below:

Sub-disciplines of the AI field

Computer hardware design, general intelligence, and cybernetics are all terms for bringing engineering and intelligence together. One of our early discoveries was that some things that we thought would require a lot of general human intelligence can actually be done easily using fairly simple tricks, and by different tricks than the ones humans use. In the beginning of the AI field in the 1950s it seemed like taking derivatives and integrals would require a lot of human level intelligence, but instead we've learned a lot about solving some problems that are hard for humans to solve by using simple tricks. We've also learned how to reasonably well work on vision, hearing, and movement using very crude, very rough approximations of parts of the brain. Through many ups and downs in the AI field through successive waves of over-optimism and over-pessimism in various parts of the world, numerous sub-disciplines of the AI field have come together in various different ways.

Neural nets is one sub-discipline of the AI field that has been very popular in the last few years (Fig. 27). I personally

find this very ironic. I was looking for a PhD topic in 1987 or so, and wanted to do a PhD on recurrent neural nets to model dynamic processes, but no one wanted to supervise that topic. I did research on that and other parts of AI for many years after that, and now people think neural nets are everything, and you are stupid doing any other types of AI. I tend to ignore the wild swings of opinion, but I am impressed by some of the recent successes of deep neural nets and shallow neural nets. I think these are probably tools for carrying out certain intelligent functions, but they also have some limitations.

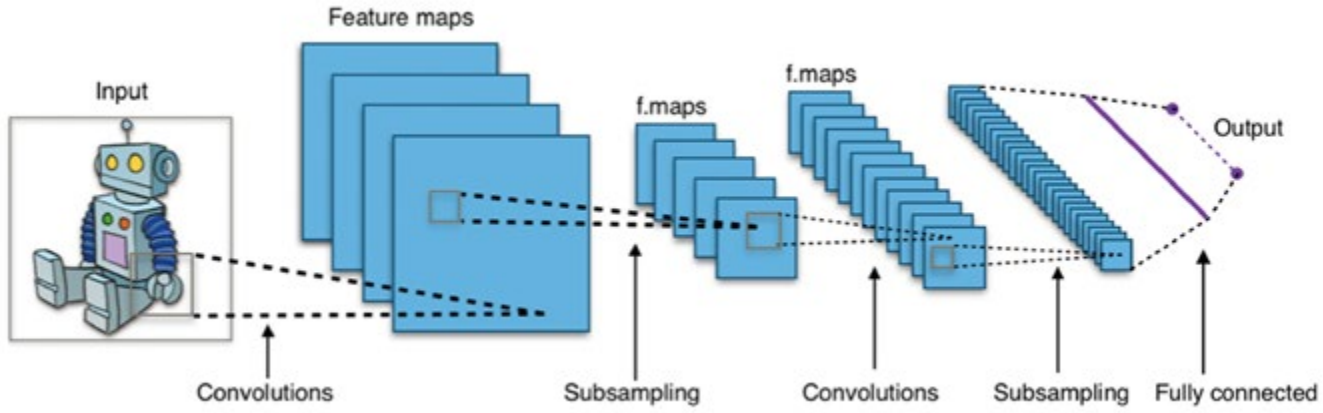


Figure 27. *Convolutional Neural Network. "In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery."* <https://en.wikipedia.org/wiki/Convolutional_neural_network>
This work has been released into the public domain by its author, Aphex34. I, the copyright holder of this work, hereby publish it under the following license: This file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license.
Source: https://en.wikipedia.org/wiki/File:Typical_cnn.png

Evolutionary learning is another discipline of AI, going back to the 1970s, and simulating natural selection in various ways in the computer. I was really happy to get John Koza to come to this workshop. He has been one of the super, uber pioneers of evolutionary learning, pioneering genetic programming (see Fig. 1), the evolution of computer programming. I think this is a discipline of AI that is particularly relevant to the topic of this workshop, which I will talk about a little bit later, because DNA is especially useful for design, for designing complex systems, and much more so than deep neural nets, as an example, although I think those have a big role to play in AI for nanotechnology also.

Another area of AI that has a lot of significance for logic systems is the work of John McCarthy, Marvin Minsky, and others in the '60s, and again this has had various ups and downs, and logic systems have had some some successes that have had some relevance to design. We use logic systems to design and check circuits and instruction sets, etc. for computers. What has been found is that to do logic-based AI in a very general, domain-independent way is an NP-hard problem. If you can solve that, you get a new level of thinking. No one has done that yet. But in any specific domain you can figure out ways to guide a logical examination process to make a logic-based AI useful. Really what happens inside Mathematica or Maple is logic-based AI, but we don't call it AI because it works. They called it AI in the 1950s when they were figuring out how to get it to work. There have been a lot of successes there and in certain other domains also. When I started as an undergraduate in math, you could not go to a computer and have it solve a differential equation for you. Now you can, which is exciting.

There is a lot of statistical learning in AI, which isn't neural networks, but there are many of the same functions crafted in support vector machines, and kernel methods and decision rules and so forth. Where is the border between advanced statistical learning and AI is not especially clear. If you have a supervised learning problem, and you solve it with support vector machines, I guess you are doing statistics. If you solve it with a multi-layer perceptron, then maybe its AI. If you solve it with a deep neural network, then you are doing AI.

We have a discipline in AI called cognitive architectures. Here the goal is to make a system that emulates all the key parts of the human mind, like declarative memory, episodic memory, semantic memory, procedural memory, procedure of learning, practical learning, learning about self and others, etc. In this sort of AI one is not as worried about learning complex patterns—although you do want to do that—the goal is that if you get the architecture of the mind down first; then, once you've done that, the learning algorithms turn out to be innate. This also has a long history, going back to the General Problem Solver of Newell and Simon in the '70s, and SOAR and ACT-R and a number of historical problems of architecture, and a whole bunch of new systems developed along those lines.

My own AI work was OpenCog, an open source AI platform. We've taken an integrated approach. We have a certain cognitive architecture into which we've plugged different AI algorithms from different paradigms. We are not the only ones to take that kind of integrated approach either. We have a probabilistic logic that we use for more abstract reasoning and learning, a deep neural net for visual and auditory perception, a probabilistic genetic programming for creating new concepts, and a neural net-like activity to spread attention through the system. That is one approach to these multiple paradigms: to find the theoretical framework in which they can all work together.

Relation of AI to Neuroscience

So that is an overview of different parts of AI as they exist now. Before I get into a few minutes on how that can apply to nanotech, I should address the relation of AI to neuroscience, which often comes up, especially since there are a lot of chemical and biological scientists here. Certainly neuroscience has been very inspirational for AI throughout its history, and some of the more successful AI methods are at some level approximations of what happens in some parts of the brain. On the other hand, I've also done computational neuroscience for a few years to model specific circuits in the parietal cortex that we use for selecting actions. What we did in computational neuroscience bore very little resemblance to what we do in AI. We were modeling neurons by pretty complex, nonlinear differential equations rather than by the simple, formal neurons that you have in neural net models. With all these neurotransmitters and glial cells, there is a lot going on in neurons that we do not understand yet, and there is a lot going on that we do sort of understand, that does not seem to be in any computer science neural net models. That would be a whole topic unto itself.

The point to be made is that computational neuroscience is a very interesting topic, and there are some AI models that were loosely inspired by aspects of the brain, which is important, but what you find is that the most algorithmic success and practical success has been obtained by taking models loosely inspired by the brain, and making registration looser and looser, rather than tighter and tighter. For example, in deep neural nets Tomaso Poggio is a great neuroscientist who had some hierarchical deep neural net models for computer vision that were great. However, how you obtain higher accuracy is by making things much less biologically realistic, and instead making rigidly structured convolutional neural nets, adding information theory into the mix, and so forth.

The brain as an inspiration is great, but I don't think we can look at it as a detailed design template. That is even

more important when you start to think about AI for designing nanosystems. It is not surprising that the human brain is not very good for designing nanosystems, because that is not what our brain evolved for. We are instead evolved for face recognition or for walking down the street. You would expect that more rather than less deviations from the human brain would be useful for designing nanosystems.

Role of AI in Nanotechnology Research

What do we have in AI today that is useful for helping in particular in nanotech, but also in any science? It breaks down into a number of different capabilities, which is consistent with the fragmentary nature of the AI field at the moment.

Probably the number one practical use right now is data analysis. Whenever you have complex data, and the relevant patterns in the data involve complex combinations of multiple variables, AI—used appropriately—is likely to do better than any other method. That has been demonstrated in many different domains: computer vision, analyzing genomic data, chemical data, economic data etc.

Another area is system design. It is more engineering than science. It weaves together designing complex systems with very particular and complex constraints. It is very difficult for the human mind to balance all of these together.

There is natural language processing. There is way too much data and too many papers for any of us to read and process all of the relevant information. To the extent that AI can look at a lot of information and summarize it and find the deficiencies that are relevant, that would be useful. That task is really hard in science because the papers are complicated. Probably the most promising area is the biomedical domain.

There has been some work, but not enough, on automated discovery of laws of nature. You are not just trying to learn a highly specific data pattern; you are trying to learn broadly applicable rules. There have been some cool prototypes, but no huge successes in this effort. If you feed in the right data, AI can deduce the natural law. But how did you know what was the right data? Because you knew that natural law! This is fairly early stage, but it is interesting and worth mentioning.

Then we have theorem proving and automated inference. By and large they are problems in mathematics, but there are similar problems in biology where you take a bunch of formal equations characterizing a biology model, and you use an AI to derive logical conclusions to those equations. These become new hypotheses, which you can test by experiment, and you can do deductive, inductive, and abductive reasoning trying to draw conclusions from formalized knowledge, which is either encoded by people or abstracted from data by some rule induction system.

So we have a lot of different tools here, and there are a lot of different ways to apply those in the context of nanotechnology. A little later today I am going to give another brief spiel on one way I think these tools could be applied to nanotechnology, which does involve integrating various particular AI tools in a particular way, but there are surely a lot of other interesting ways to apply AI to nanotechnology. So part of my agenda here is to get help from people with expertise in nanotech in responding to the AI sketch I have been thinking of. Another agenda item is just to brainstorm and get more cool ideas on how to apply these AI tools to nanotech.

To briefly encapsulate a couple of the ideas:

■ People have applied neural nets to simulate a physics engine. You have a physics engine in a video game where it simulates rigid body newtonian mechanics. You turn to a neural net to simulate what that physics engine does, but then you allow other examples, and then it does much faster physics simulation, and after running the` simulation [you have a better result]. What if you did that for a carbon chemistry simulator? It is a little harder, but it should be very interesting to get a deep neural net that actually ran a quantum chemistry simulation. Mathematically it should be possible. There would be more variables, so you would have to run more examples. There would be a lot of challenges, but there is no objection to that in principle.

■ Genetic Programming for designing nanoscale systems. John Koza wrote four fat books on GP, and a couple of them are just about designing systems. There is also some work by Spector on using genetic algorithms to design quantum circuits[11]. There the AI is coming up with combinations of unitary matrices to design quantum circuits. What was disappointing is that he did not, as far as I can see, discover any new quantum algorithms. He discovered the same four algorithms that we all knew about. But it is interesting that he did that, so along those lines, evolutionary learning should be able to design nanosystems. Quantum circuits are nanosystems, so it should be able to design nanomachines and nanocomputers, and so forth. Of course, this ties into using deep learning to simulate a quantum chemistry simulator because evolutionary learning requires a fitness function. If your fitness function is a slow simulator, that is bad. If your fitness function is an accelerated simulator, that is good.

■ You can see then the involvement in other approaches you could take. For example, you can apply a theorem prover to the laws of physics relevant to nanosystems, and perhaps you estimate that a certain nanodesign that resulted from Genetic Programing is bad, because it won't work because of what would be just common sense based on equational reasoning based on interpreting a sequence of actions that AI predicted in terms of the laws of physics.

Introduction to Molecular Nanotechnology:

Extending First Principles Based Methods to Predicting Nanoscale Design of Molecular Machines

William A. Goddard III
Dept. of Chemistry, Materials Science & Applied Physics, Caltech

Please find the transcript of William Goddard's introduction to molecular nanotechnology at the AI for Atomic Precision Workshop below. His presentation slides can be found here.

From the Atomic Scale to the Macroscopic Scale

What I want to do today is give an overview of what is going on starting at the other end. We would like to start with fundamental principles and predict interesting systems, going from quantum mechanics to reasonably large-scale objects of 20 million atoms. What I want to talk about here are data and concepts. You can analyze data all you want, but what you have to get is the concepts. Once you have the concepts, you can do things with them.

The Grand Vision has been, at least in my mind since the 1970s, to use theory to actually predict and design things. It's a beautiful concept, and we know it is eventually going to happen. Probably I was too early to think it was going

[11] "Machine invention of quantum computing circuits by means of genetic programming" Lee Spector, Jon Klein. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 22, 275–283 (2008)

to happen in the 1980s, but it is happening now.

To design new materials, it is critical to have the highest quality first principles quantum mechanics to get enough data to go on. But you have to take that data and get concepts, and then extend those concepts to doing chemically reactive processes. If you are going to synthesize materials, you have to do that at least at the electrochemical range of one or two million atoms to have enough of a nanoscale that you can develop appropriate concepts. That is what we have been focusing on.

To use computational modeling of materials and processes as the basis for in silico design for catalysis, materials, pharmaceuticals, energy, and nanotechnology it is necessary to start with quantum mechanics methods, but to use hierarchies of overlapping, experimentally validated, computational models to connect the atomic scale to the mesoscopic and macroscopic scales. This goal thus encompasses a distance scale from Ångstroms to nanometers, to microns, to millimeters and beyond, and a time scale from femtoseconds, to picoseconds, to nanoseconds, to milliseconds to hours.

Improving Molecular Dynamics Simulations with Metadynamics

My view is that we have new methods to do quantum mechanics. The strategic question here is to be able to do metadynamics to get free energies at room temperature. But quantum mechanics is limited to about 300 atoms, which is not a large system. We need model systems where catalysis and electrochemistry let us synthesize very interesting materials. A very important development is to extend quantum mechanics accuracy from 300 atoms to 3 million atoms. We have been working on this for the last 15 years and it has actually been working very well. It is extending quantum mechanics to doing quantum descriptions on millions of atoms at high temperatures. We are actually making major progress in predicting the structures of membrane proteins and predicting how they activate.

We can also extract properties of systems, like free energies, using approaches now a thousand times faster than old-fashioned numerical integration approaches. The challenge is how to connect first principles quantum mechanics to macroscale systems so that we can predict new materials before they are synthesized. We use an overlapping hierarchy of methods. This is over-simplified. There are hundreds of methodologies.

Today I will discuss four advances enabling multiscale simulations:

- New Methods of Quantum Mechanics (QM), now extended to do electrocatalysis as a function applied potential
- ReaxFF[12,13] reactive force field for near QM accuracy in describing reaction barriers but with computational costs of force field Molecular Dynamics (MD), enabling reactive simulation of systems with millions of atoms, and extending time scale simulated from 100 nanoseconds to seconds
- 2PT methodology for extracting entropy and free energy along MD trajectory at no extra cost. Can be used with millions of atoms

[12] "ReaxFF: A reactive force field for hydrocarbons" ACT van Duin, S Dasgupta, F Lorant, WA Goddard III. J. Phys. Chem. A **105**, 9396-9409 (2001). PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/471.pdf

[13] "Prediction of structures and properties of 2,4,6-triamino-1,3,5-triazine-1,3,5-trioxide (MTO) and 2,4,6-trinitro-1,3,5-triazine-1,3,5-trioxide (MTO3N) green energetic materials from DFT and ReaxFF molecular modeling" S Naserifar, SV Zybin, CC Ye & WA Goddard III J. Mater. Chem. A, DOI: 10.1039/C5TA06426K (2015).
PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/1150.pdf

■ eFF: nonadiabatic QM on highly excited electrons, including electron ejections, and plasma interactions for millions of electrons

We decide what to do our theory on by the applications that we think are relevant to solve. We work in lots of areas, always on a problem that is critical, that we think we can solve, that has not yet been solved. We worked for a long time on difficult industrial problems that have forced the development of new methods. We have spun off a few software companies in the past, but now our philosophy is to give our software away and let other people update it. I will talk a little bit about how we use our force field and what we have done.

Predicting the structures of membrane proteins and figuring out how they are activated by their ligands: By making first principles improvements to molecular dynamics methods, we have made major progress with predicting the structures of membrane proteins and actually figuring out how they are activated by ligands. First principles structure prediction of GPCR (G-Protein Coupled Receptors) and studies of the ligand activation of GPCR have been made possible by molecular dynamics studies using newly developed methods and force fields[14,15,16,17].

Reduction of CO2 on a copper surface: We use quantum mechanics in a form that we can get free energy barriers in the time scale of 10 or 20 picoseconds. I won't go into details. The fundamental idea goes back to Laio and Parrinello[18], but we were the first to apply it to actual real industrial problems[19,20,21]. We judge the system by how well we can do, how accurate are our methods: electrochemistry for CO2 reduction and oxygen reduction. There are high quality data to compare it with. We are very pleased to find out that when we used these methods for onset potential for the various parts of CO2 reduction, from CO2 to organics that we are within about 0.05 eV and get the

[14] "Molecular mechanisms underlying differential odor responses of a mouse olfactory receptor" WB Floriano, N Vaidehi, WA Goddard III, MS Singer, GM Shepherd. Proc. Natl. Acad. Sci. USA **97**, 10712–10716 (2000). PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/443.pdf

[15] "First principles predictions of the structure and function of g-protein-coupled receptors: validation for bovine rhodopsin" RJ Trabanino, SE Hall, N Vaidehi, WB Floriano, VWT Kam, WA Goddard 3rd. Biophysical Journal **86**, 1904–1921 (2004). PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/555.pdf

[16] "Prediction of the 3D structure of FMRF-amide neuropeptides bound to the mouse MrgC11 GPCR and experimental validation" J Heo, S-K Han, N Vaidehi, J Wendel, P Kekenes-Huskey, WA Goddard, III. ChemBioChem **8**, 1527-1539 (2007). PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/720.pdf

[17] "The predicted binding site and dynamics of peptide inhibitors to the methuselah GPCR from Drosophila melanogaster" J Heo, WW Ja, S Benzer, WA Goddard III. Biochemistry **47**, 12740-12749 (2008). PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/771.pdf

[18] "Escaping free-energy minima" A Laio & Michele Parrinello. Proc. Natl. Acad. Sci. USA **99**, 12562–12566 (2002), doi: 10.1073/pnas.202427399 http://www.pnas.org/content/99/20/12562.full.pdf

[19] "Free energy barriers and reaction mechanisms for the electrochemical reduction of CO on the Cu(100) surface including multiple layers of explicit solvent at pH 0" T Cheng, H Xiao & WA Goddard III. J Phys. Chem. Lett. **6**, 4767–4773 (2015), DOI:10.1021/acs.jpclett.5b02247. PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/1145.pdf

[20] ""Mechanistic Explanation of the pH Dependence and Onset Potentials for Hydrocarbon Products from Electrochemical Reduction of CO on Cu(111)" H Xiao, T Cheng, WA Goddard III & R Sundararaman. J. Am. Chem. Soc. **138**, 483–486 (2015). DOI: 10.1021/jacs.5b11390. PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/1149.pdf

[21] "Reaction Mechanisms for the Electrochemical Reduction of CO2 to CO and Formate on the Cu(100) Surface at 298 K from Quantum Mechanics Free Energy Calculations with Explicit Water" T Cheng, H Xiao & WA Goddard III. J. Am. Chem. Soc. **138**, 13802–13805 (2016) DOI: 10.1021/jacs.6b08534 http://pubs.acs.org/doi/abs/10.1021/jacs.6b08534

right product (Fig. 28). That is very encouraging. I did not think the calculations were that accurate; they may not be for all systems. but we have good case examples.

Similarly, we can do dynamics of reactions in the presence of solvents and again we get free energy barriers to within 0.05 eV. This reaction of reducing dioxygen to water in fuel cells is the last step. It transfers a proton to this OH on the surface. It has a chain of molecules involved. You have to have a surface to do that. It has 300 atoms and takes 20 picoseconds per case.

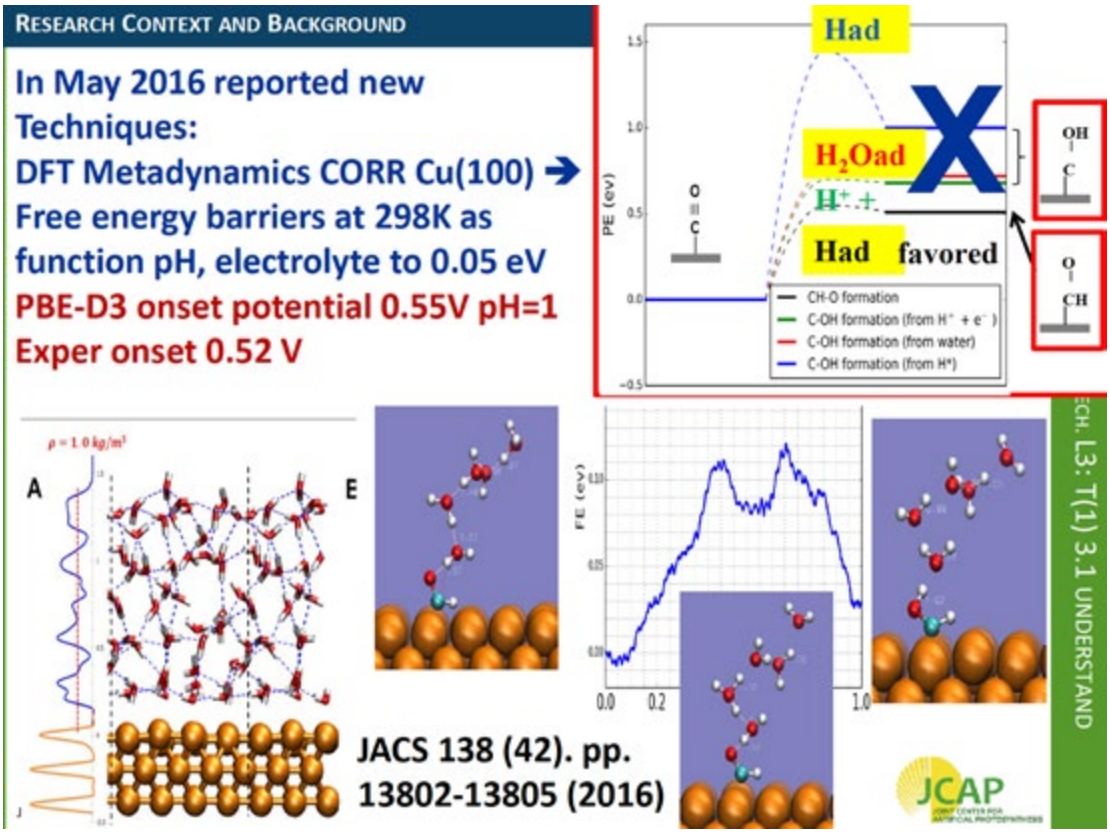


Figure 28. *Use of DFT Metadynamics to reveal reaction mechanisms for the electrochemical reduction of CO2 to CO and formate from Quantum Mechanics to calculate free energies and explicit 5 layers of water molecules in molecular dynamics. The goal is to use QM to determine mechanism and then to use mechanism to design better catalysts than the Cu(111) surface (see footnote 20 above). Credit: William A. Goddard III, used with permission.*

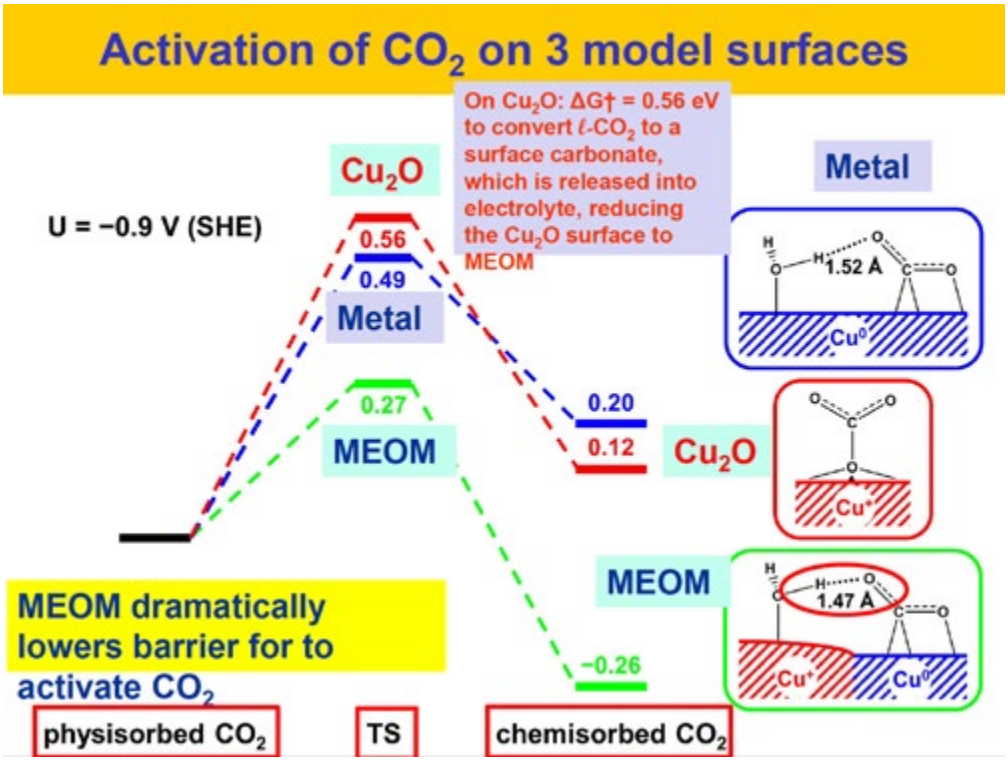
These results provide evidence for the paradigm that inspires our work: (1) use quantum mechanics to determine kinetic mechanism and free energy change at 298 K (approximately room temperature), and then (2) use mechanism to design new catalysts.

MEOM (Metal Embedded in Oxidized Matrix) Catalysis

The catalysts in these systems are not anywhere near good enough. The lack of good catalysts inspired new ideas. People had in mind a metal as a catalyst to convert CO2 to organics; there were reports that oxides could also catalyze this reaction. But we found that the oxide itself does not do it. If you remove some of the oxygen from the surface, you leave a mixture of a metallic region and an oxide region. At the interface between those two regions you can stabilize some of the CO2, and to make CO you can stabilize the formation of C2 bonds. The catalyst is actually a Cu metal embedded in oxidized matrix (MEOM[22] and Fig. 29). That raises a concept; now we use the concept to define the system, and we use the same concepts to advise our experimental friends on the problem of synthesis. You use the concepts for synthesis to make the material.

Summarizing the results found with MEOM catalysis:

- The major two-carbon products from the CO2 reduction reaction are either ethylene (at neutral pH) or ethanol (at basic pH).
- We showed that the energetics of surface water determines the selectivity of alcohol vs hydrocarbon products.
- At neutral pH surface water donates a proton for dehydroxylation to form hydrocarbon products.
- In basic pH the ability of surface water to dehydroxylate the surface species is suppressed (because the product OH− is less favorable), favoring instead the alcohol product (ethanol).



[22] "Cu metal embedded in oxidized matrix catalyst to promote CO2 activation and CO dimerization for electrochemical reduction of CO2" H Xiao, WA Goddard III, T Cheng, Y Liu. Proc. Natl. Acad. Sci. USA **114**, 6685–6688 (May 9, 2017) doi: 10.1073/pnas.170240511 http://www.pnas.org/content/114/26/6685.full.pdf

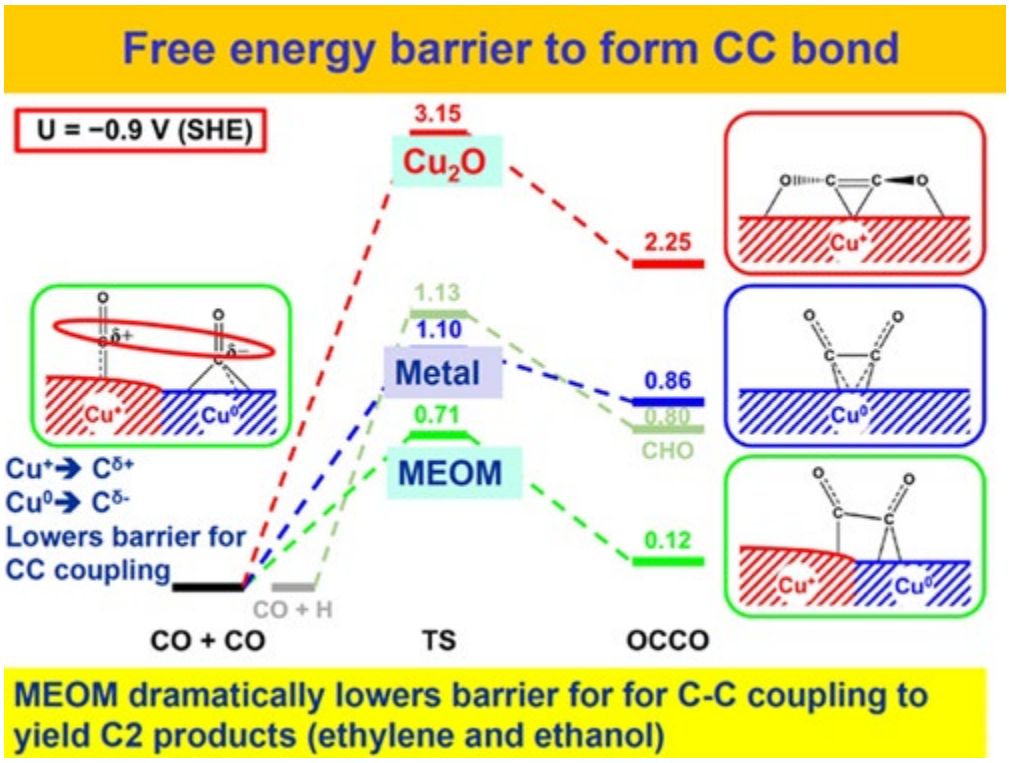


Figure 29. *The electrochemical reduction of CO2 is catalyzed by Cu metal embedded in oxidized matrix (MEOM, Ref. 11). (A) Catalysis by MEOW lowers the energy of activation more than catalysis by either Cu metal alone or CuO alone. (B) Catalysis by MEOW lowers the free energy to CC bond formation more than catalysis by either Cu metal alone or CuO alone.* Credit: William A. Goddard III, used with permission.

The MEOM model demonstration that the most active surface is only partially oxidized indicates that active surface Cu+ alone do not improve the efficiency of the carbon dioxide reduction reaction, and instead the synergy between active surface Cu+ and metallic Cuo regions in the MEOM catalyst improves the kinetics and thermodynamics of both carbon dioxide activation and carbon monoxide dimerization. The model suggests alternative oxidized matrices, like silver sulfide, might deliver similar electrostatic contributions while leading to much improved electrochemical stabilities.

Grain Boundaries and Reactive Force Fields

Although we got some interesting results from single crystals, the MEOM results above provide an indication that the interesting catalytic systems are not single crystals. It came out recently that copper nanoparticles are much better than single crystal copper electrodes[23]. Experimentally it was found that the grain boundaries of the nanoparticle were binding the CO2 more strongly, and that was somehow correlated with extra activation. This result

[23] "A Direct Grain-Boundary-Activity Correlation for CO Electroreduction on Cu Nanoparticles" X Feng, K Jiang, S Fan, MW Kanan. ACS Cent. Sci. **2**, 169–174 (2016) DOI: 10.1021/acscentsci.6b00022 http://pubs.acs.org/doi/abs/10.1021/acscentsci.6b000 [Open Access]

implies that grain boundaries are responsible for creating the vast majority of active surfaces. Another experimental study[24] revealed that oxide-derived copper electrocatalysts prepared by hydrogen reduction of cuprous oxide showed surface sites that were active in carbon monoxide reduction, and that this electroreduction activity correlated with metastable surface feature that strongly bind carbon monoxide. These experimental results indicated that specific structures associated with grain boundaries were very active catalysts. Simulations should be very useful in uncovering such site, but to simulate the system takes about 200,000 atoms to get a reasonable description of a nanoparticle about 10 nm thick and 26 nm long. Such a simulation would be far too large for DFT (QM) methods.

We've spent 15 years developing a method to use quantum mechanics for large-scale systems. It's called reactive force fields. Twenty years ago we developed a very simple way of predicting charges, but we did not figure out a simple way to test it until very recently. We found out that this is very accurate, just as accurate as quantum mechanics. Combining this with reactive force fields (see footnotes 11 and 12 above) we can do a system of a million atoms, with all kinds of defects and interfaces, and we can do it at 300 degrees C.

To mimic the experimental results reported with Cu nanoparticles and oxide derived copper electrocatalysts, we did a 201,755-atom simulation growing a Cu nanoparticle in the presence of a 10 nm carbon nanotube, and then annealed it for tens of nanoseconds. The simulation revealed a proportion of active CO binding sites similar to the experimental results. When we analyze grain boundaries, about half the sites that bind CO strongly have low barriers to making C2 bonds. Looking at the sites that work well, it turns out that the most useful kinds of surface sites are stacking faults. We use quantum mechanics to get confidence in fundamental reaction time steps, then we extend that to large scales using reactive force fields to be able to simulate systems more like what the experimentalists make, and we have now determined the site that is important. The sites identified as active suggest synthetic pathways to more controllable catalytic sites.

The simulated structures also showed similar grain boundaries to the experimental transmission electron micrographs of Cu nanoparticles. Two good grain boundary sites have a square stacking fault (Fig. 30).

[24] "Probing the Active Surface Sites for CO Reduction on Oxide-Derived Copper Electrocatalysts" A Verdaguer-Casadevall, CW Li, TP Johansson, SB Scott, JT McKeown, M Kumar, IEL Stephens, MW Kanan, I Chorkendorff. J. Am. Chem. Soc. **137**, 9808–9811 (July 21, 2015) DOI: 10.1021/jacs.5b06227 http://pubs.acs.org/doi/abs/10.1021/jacs.5b06227

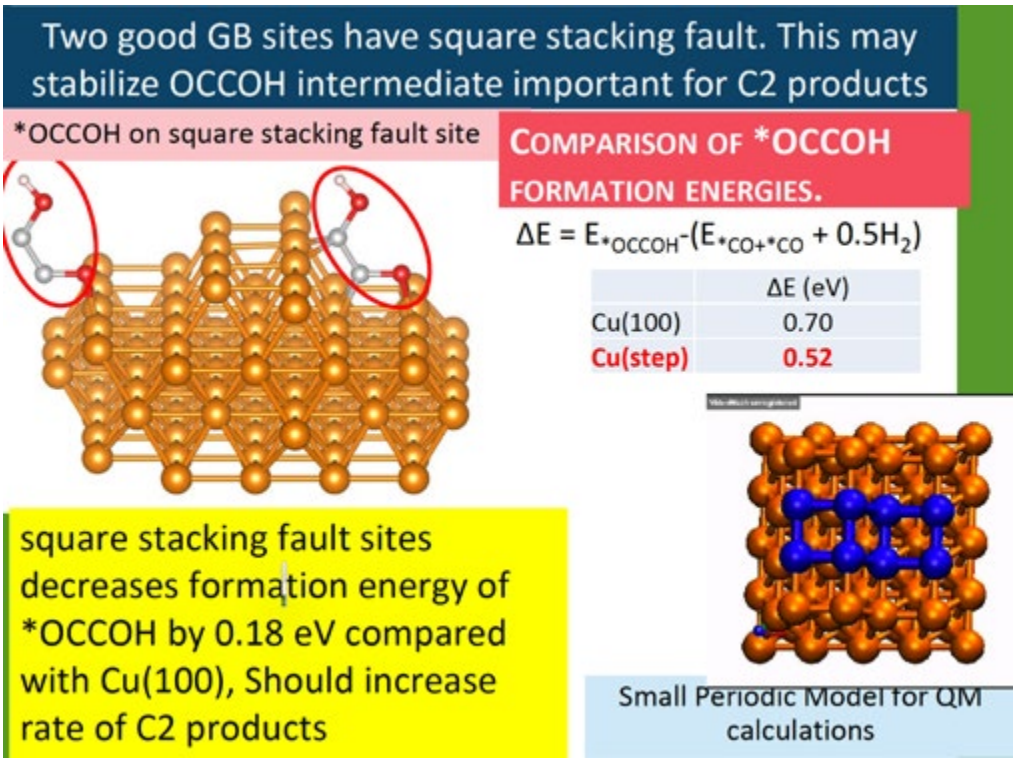


Figure 30. *Square Stacking Fault in Simulation of Cu Nanoparticle Grain Boundary. Credit: William A. Goddard III, used with permission.*

In summary, we used the ReaxFF Reactive Force field (fitted to DFT QM) to grow a 20 nm Cu nanoparticle (200,000 atoms) on a 10 nm carbon nanotube to mimic experiments. X-ray diffraction and Transmission Electron Micrograph simulations show the similarity between our predicted nanostructure and the experimental structure. DFT calculations on less than 300 atoms were carried out to probe the active sites from the ReaxFF nanoparticle simulation using binding energy as the criterion. We found specific surface sites that lead to high performance for CO reduction to two-carbon products. We propose a simple structure to explain improved CO reduction performance (Fig. 31). We tested the new site using QM, and found it is 55 times more active.

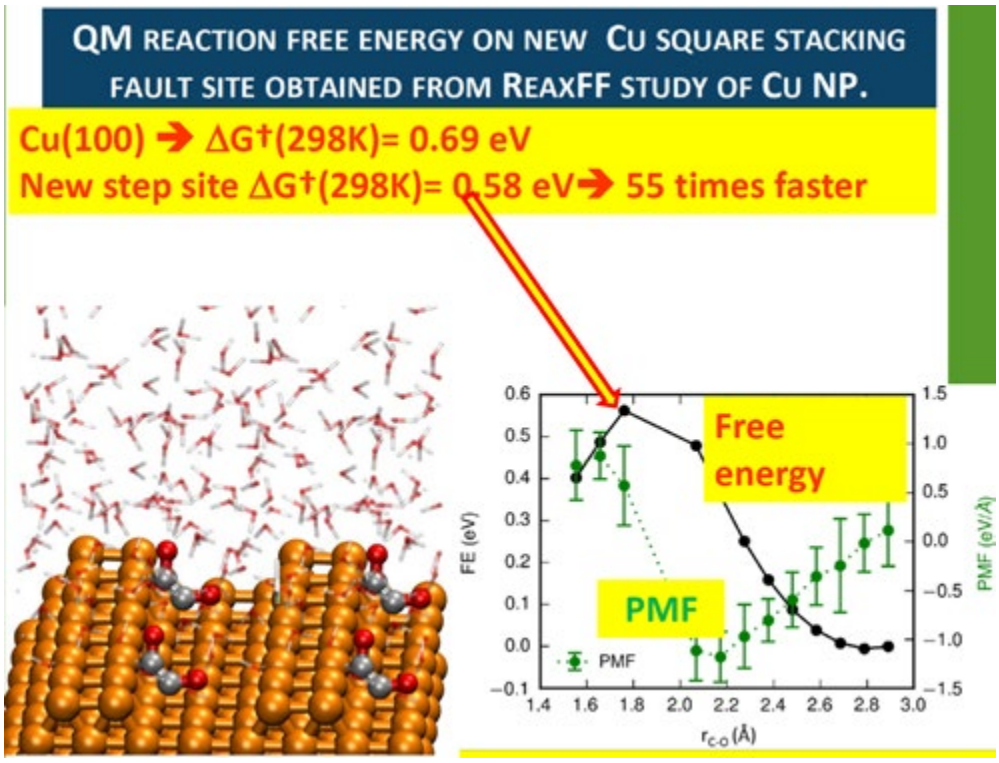


Figure 31. *Quantum mechanical free energy calculated for the Cu square stacking fault site obtained from the ReaxFF study of the Cu nanoparticle indicates it is 55 times faster.* Credit: William A. Goddard III, used with permission.

Jagged Surfaces Make Platinum 50 Times more active catalytically

Additional evidence for increased catalytic activity of some irregular structures comes from work we did with colleagues at UCLA on the oxygen reduction reaction catalyzed by platinum[25]. The experimentalists began with core-shell nanowires composed of a NiO core and a Pt shell. Annealing these left a nickel-platinum alloy with 70 % Ni. Dissolving all the Ni left a pure platinum nanowire catalyst with a jagged, irregular surface, as determined by transmission electron microscopy. Electrochemical characterization revealed platinum 50 times more active than the best current platinum carbon catalyst. And it's 15 times better than the 3M catalyst that is touted to be so great.

To gain insight on why these J-PtNWs could deliver such an increase in oxygen reduction reaction activity, we conducted reactive molecular dynamic studies using the ReaxFF to simulate leaching Ni atoms to form J-PtNWs, followed by a local optimization to predict Pt-Pt distances (Fig. 32). We concluded that "Reactive molecular dynamics simulations suggest that highly stressed, under-coordinated rhombus-rich surface configurations of the jagged nanowires enhance ORR activity versus more relaxed surfaces." Now we are trying to figure out which of these sites is doing the chemistry. We have not figured that out yet.

[25] "Ultrafine jagged platinum nanowires enable high platinum mass activity for the oxygen reduction reaction" M Li, Z Zhao, T Cheng, A Fortunelli, C-Y Chen, R Yu, Q Zhang, L Gu, B Merinov, Z Lin, E Zhu, Q Zhang, T Yu, Q JiaJ Guo, L Zhang, WA Goddard III, Y Huang, X Duan. Science **354**, 1414-1419 (16 Dec 2016) DOI: 10.1126/science.aaf9050 http://science.sciencemag.org/content/354/6318/1414

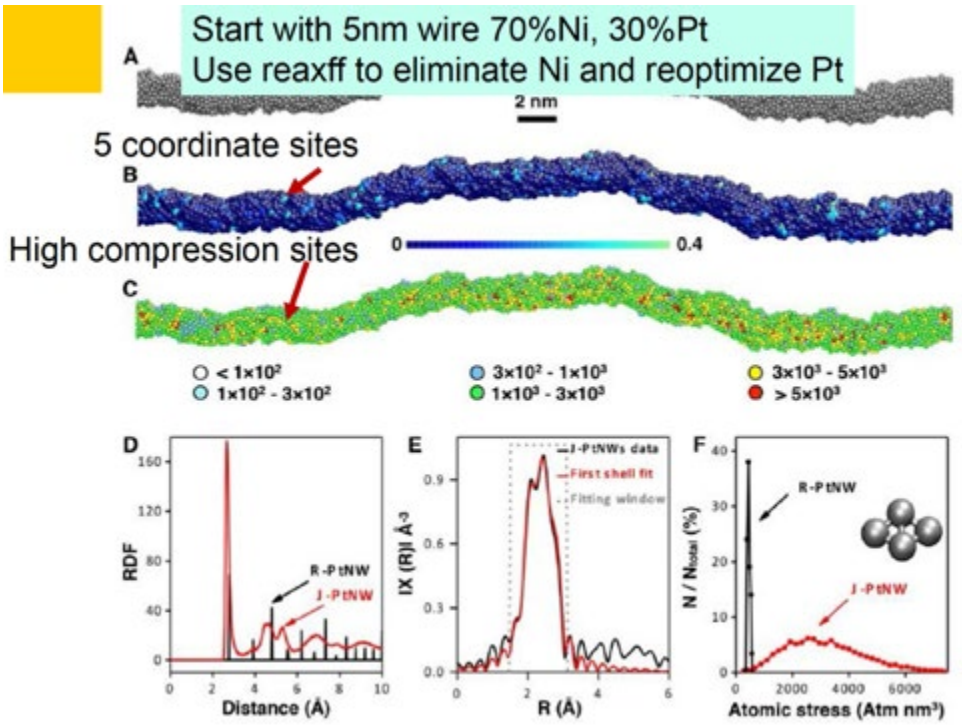


Figure 32. *Structural analysis of the J-PtNWs obtained from Re axFF reactive molecular dynamics and x-ray absorption spectroscopy. (A) Pictorial illustrations of the final structure of a Pt J-NW generated by Reactive Molecular Dynamics simulations, with an average diameter of ~2.2 nm and length of ~46 nm. (B) J-PtNW with colored atoms to show the 5-fold index. (C) JPtNW with colored atoms to show distribution of atomic stress (in atm·nm3). (D) Pt-Pt radial distribution function (RDF) of the SMA-predicted J-PtNW (red) compared with the peaks of the RDF for the regular PtNW (black). (E) Pt L3 edge FT-EXAFS spectrum (black) collected ex situ and the corresponding first shell least-squares fit (red) for the J-PtNWs. (F) Distribution of the absolute values of the average atomic stress on surface rhombi for the R-PtNWs (black) and the JPtNWs (red). A rhombus is an ensemble of 4 atoms arranged as two equilateral triangles sharing one edge as shown in the inset."* Credit: William A. Goddard III, used with permission.

Accelerated Molecular Dynamics

We have developed a method of speeding up simulations. Since molecular dynamics simulations do time steps every femtosecond, you are limited to how long you can continue the simulation, how many steps you can simulate. The accelerated dynamics approach allows us to go much faster, as shown by Fig. 33. For this particular system, the oxidation of molecular hydrogen, we have been able to go all the way from ignition temperature (798K) to flame temperature (2998 K). This simulation demonstrates that the time to convert half of the reactant H2 to product H2O is 538 seconds, but the computational time required for the simulation is only 1.3 ns[26]. I am very excited about that. This is a decrease of half-a-trillion-fold in the computational cost of molecular dynamics simulations. Thus ReaxxFF technology has been extended to address the remaining issue in first-principles-based simulations of reaction dynamics of complex systems.

[26] "Adaptive Accelerated ReaxFF Reactive Dynamics with Validation from Simulating Hydrogen Combustion" T Cheng, A Jaramillo-Botero, WA Goddard, III, H Sun. J. Am. Chem. Soc. 136, 9434- 9442 (June 2, 2014) DOI: 10.1021/ja5037258 http://pubs.acs.org/doi/10.1021/ja5037258 PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/1059.pdf

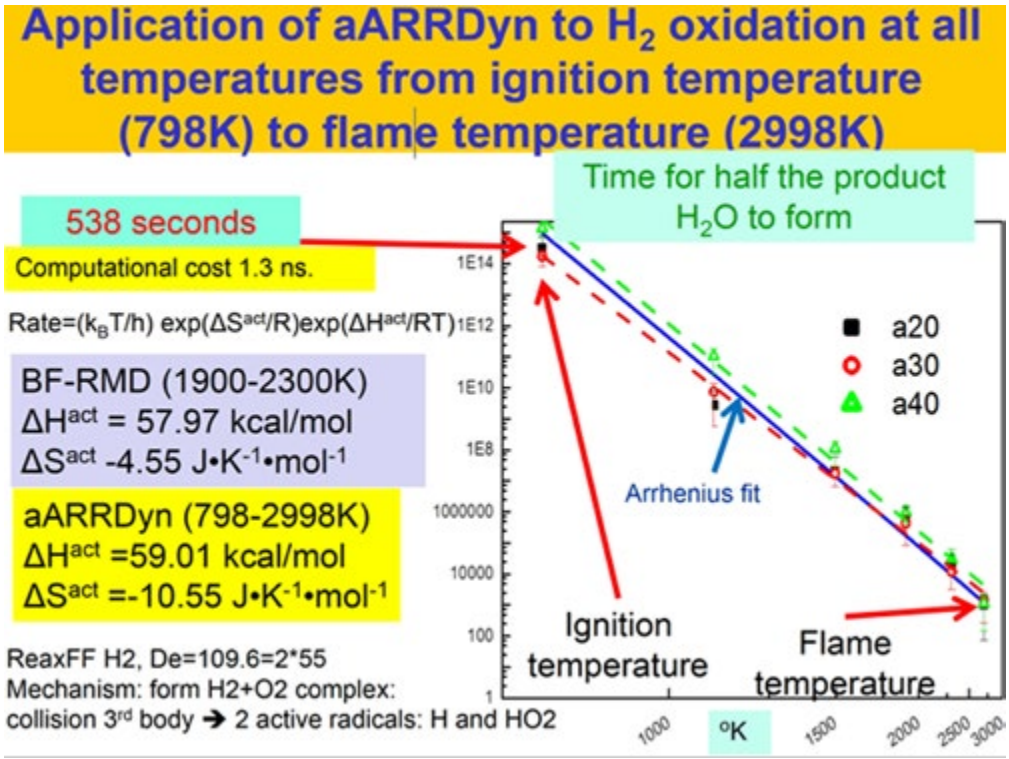


Figure 33. *Adaptive Accelerated ReaxFF Reactive Dynamics to H2 oxidation from ignition temperature (798 K) to flame temperature (2998 K). 538 seconds of reaction dynamics was simulated at a computational cost of only 1.3 nanoseconds, an improvement of half a trillion over conventional methods.* Credit: William A. Goddard III, used with permission.

Atomistic Mechanisms for Rotaxane Electronic Switches

Over the years we have done a lot of things with nanomachines, designing materials atomistically and building systems, but with no way to actually build the systems experimentally, all we could do is predict the properties. We won't talk about these systems now.

Over the years we've also had collaborations with people who actually make systems. It started with the rotaxane systems that have been developed into electronic switches that can be controlled electrically or by oxidation and reduction reactions (Fig. 34). The groups of Heath and Stoddart had designed rotaxanes, in which a ring molecule encircles a dumbbell shaped molecule that functions as a programmable electronic switch27. The ring can localize over either of two regions along the central part of the dumbbell: the tetrathiafulvalene group (TTF) in the left half or the 1,5-dioxynaphthalene group (DNP) in the right half (Fig. 12). Experiments show conclusively the presence of two distinct states (one with a resistance of 10-100 times the other. Applying an appropriate voltage or a suitable oxidant or reductant switches the system from one state to another.

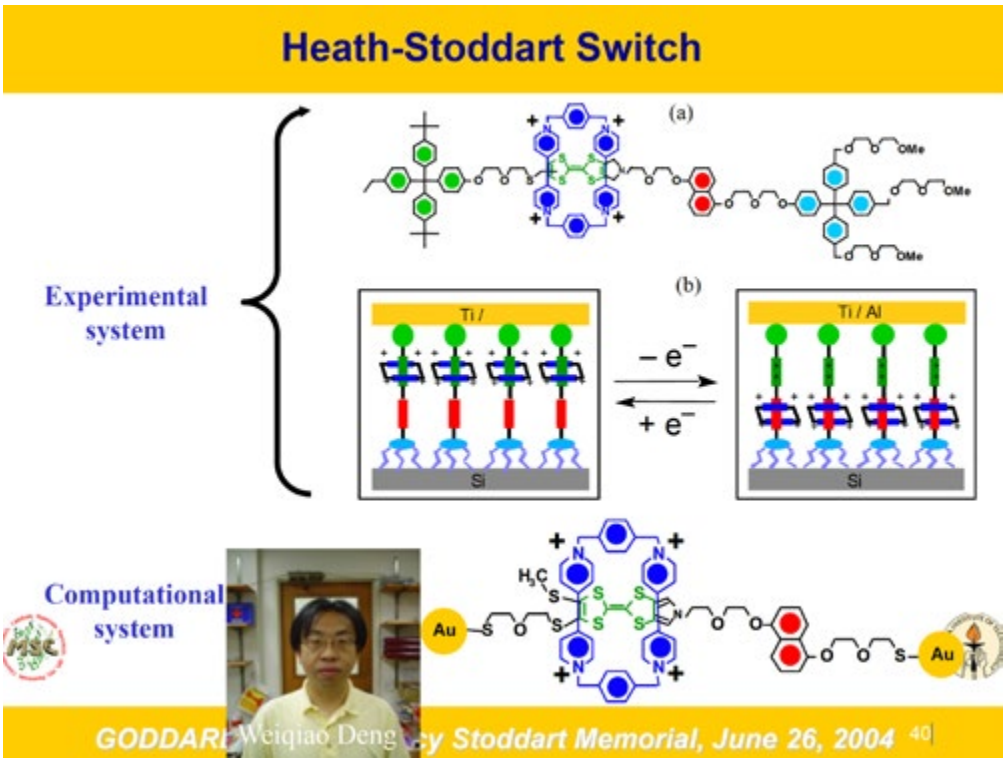


Figure 34. *(adapted from Scheme I)[27] showing at the top the structure of the Heath Stoddart switch used for experimental studies, and at the bottom, the structure of the system used for simulations. Each Au electrode consisted of 3 Au atoms. Credit: William A. Goddard III, used with permission.*

To provide the molecular level understanding to design and optimize such nanoelectronic systems, electrical conductivity at the atomistic level was calculated for the system shown in which the dumbbell end groups of the experimental system were replaced by gold atoms for calculations. The results showed large differences in conductivity between the state with the ring localized over DNP and the state with the ring over TTF. The mechanism was found to be what had been proposed by Stoddart: the ring over DNP is the ON state while the ring over TTF is the OFF state. The calculated ON/OFF ratio (37-58 times) is comparable to experimental results (10-100 times). Further, the orbital basis of the switching behavior was found to reside in the characteristics of the molecular orbitals around the Fermi energy. Specifically, the change in delocalization of the molecular orbitals affected by the ring movements was identified as playing a key role in the switching mechanism.

Electron Force Fields and Electron Enhanced Etching Dynamics

I would like to finish with one more extension of the methodology. Force fields don't have electrons; they don't have excited states. However, in many systems we want to deal with excited states. With quantum mechanics we can do highly excited states, but the motivation for developing it was to understand how low energy electron enhanced

[27] "Mechanism of the Stoddart-Heath Bistable Rotaxane Molecular Switch" W-Q Deng, RP Muller, WA Goddard III. J. Am. Chem. Soc. **126**, 13562-13563 (9/30/2004). DOI: 10.1021/ja036498x http://pubs.acs.org/doi/abs/10.1021/ja036498x PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/588.pdf

etching (LE4) works. The previous technology for etching sub-micrometer features on semiconductors depends on ion-enhanced plasma etching, which leads to pervasive damage and poor quality interfaces. The speculation was that LE4 from DC discharge, using electrons with energies below about 100 eV could be used for damage-free fabrication of features as small as 20 nm, resulting in extremely smooth surfaces. It was found that silicon structures etched by LE4 showed no damage at the crystal surface and showed dramatically reduced line edge roughness (Fig 35).

LE4 can potentially avoid the problems associated with IEE because it proceeds by a fundamentally different mechanism[28]. Because much less momentum is transferred in LE4 compared to IEF sub-surface collision cascades do not occur, so etch products are instead removed by electron-enhanced desorption. Electron stimulated desorption is stimulated by electronic excitation of quantum transitions at the wafer surface, controlled by energy thresholds specific to the wafer material, providing the opportunity to tailor the electron energy to specific materials. Momentum transfer from the electrons to the surface is negligible.

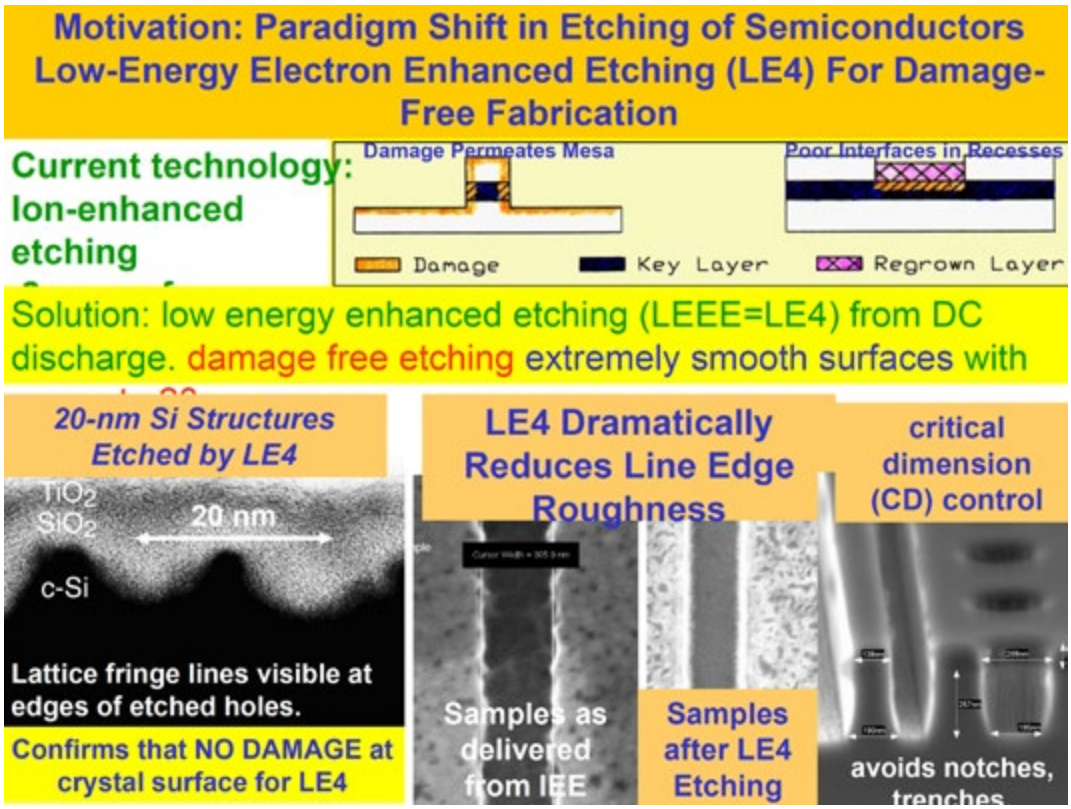


Figure 35. *Damage-free fabrication of 20-nm silicon structures made using Low-Energy Electron Enhanced Etching (LE4) instead of Ion Enhanced Etching (IEE).* Credit: William A. Goddard III, used with permission.

[28] "Precision, Damage-Free Etching by Electron-Enhanced Reactions: Results and Simulations" HP Gillis, SJ Anz, S-P Han, J Su, WA Goddard III. ECS Transactions **50**, 33-43 (2013). DOI 10.1149/05046.0033ecst. PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/1038.pdf

However, studying the dynamics of LE4 requires a force field that includes electrons, for which we developed the electron force field method (eFF)[29]. This is an approximation (shown in Fig. 36) to quantum mechanics developed to simulate the dynamics of excited states in complex materials.

We speculated that since the LE4 process produced smooth surfaces, it might be analogous to the Auger process in which an electron is ejected from a surface. We then applied eFF to study the dynamics of Auger processes in a hydrogenated diamond nanoparticle $C_{197}H_{112}$[30]. In an Auger process, the removal of an electron from an inner electron shell leads to the collapse of an electron from the higher energy valence shell into the core shell vacancy, together with the ejection of another valence electron, all occurring over several femtoseconds. The simulation reveals that nonionized but excited electrons play a key role in causing carbon-carbon and carbon-hydrogen bonds to break. Auger electrons are released from surface layers, but not from the bulk of the nanoparticle. The nanoparticle used is roughly spherical and contains six layers from the center to the surface. The Auger process occurs within the first two layers from the surface. The reason this LE4 process is so smooth is that all the etching is at the surface.

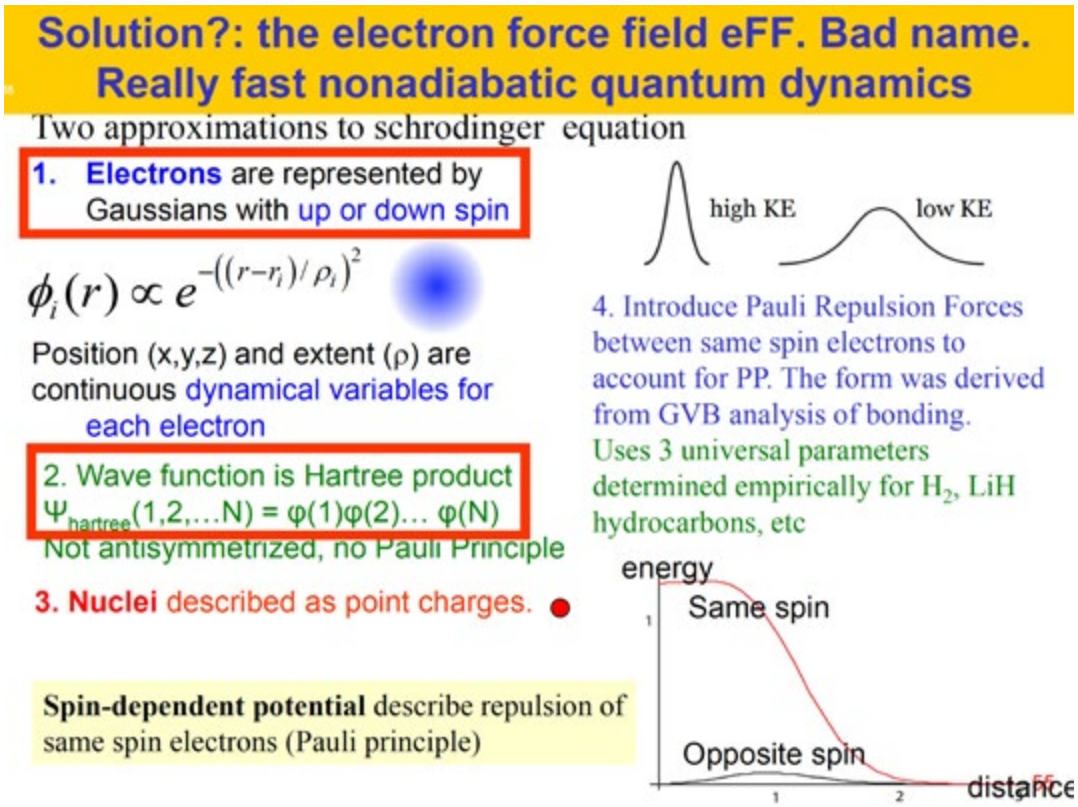


Figure 36. *The Electron Force Field: a really fast nonadiabatic quantum dynamics method. Each electron is represented by a Gaussian wave function, while nuclei are represented as point charges.* Credit: William A. Goddard III, used with permission.

[29] "Excited Electron Dynamics Modeling of Warm Dense Matter" JT Su, WA Goddard III. Physical Review Letters **99**, 185003-1 - 185003-4 (2 November 2007). DOI: 10.1103/PhysRevLett.99.185003. PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/731.pdf

[30] "Mechanisms of Auger-induced chemistry derived from wave packet dynamics" JT Su, WA Goddard III. Proc. Natl. Acad. Sci. USA **106**, 1001–1005 (January 27 , 2009). doi: 10.1073 / pnas.0812087106. PDF available on Prof. Goddard's publications page: http://www.wag.caltech.edu/publications/sup/pdf/782.pdf

Part 4: Bonus Material

After the proposal for AI for Nanoscale Design (part 1), background information on the proposal (part 2), and introductions to AI and molecular machines (part 3), this section offers bonus material from workshop conversations that contributed to exploring the intellectual landscape of how artificial intelligence could facilitate the development of atomically precise nanotechnology.

Discussion after "AI for Nanoscale Design Proposal Presentation" by Ben Goertzel and Christian Schafmeister

Audience: You mentioned functionalizing for nerve agents. What are you thinking of in terms of application?

Christian Schafmeister: Hydrolyzing nerve agents. Detoxification. Anything from spraying a tank if a bomb went off that sprays nerve agents all over the place, to spray with a catalyst that hydrolyzes it all so that it is safe to touch, to injecting it into people who have been exposes (you have a narrow window there), to prophylactically, to injecting into soldiers who might be going into a contaminated area, where the catalyst could hydrolyze the nerve agent faster than it could get into the neurons. We have support right now to do that. the Department of Defense has been putting a huge amount of money into developing enzymes that can do this, but the problem is formulating, storing, delivering enzymes is just a nightmare. They are fragile little snowflakes that fall apart when you look at them the wrong way. They unfold because proteins are fragile and recyclable. The molecules we make are pretty much indestructible and tiny. They can get into places where proteins can't. What the Department of Defense wants are abiotic, non-natural approaches to hydrolyzing nerve agents. We have a couple of ideas on how to do that. One is mimicking enzymes that hydrolyze nerve agent. Another one is taking this zirconium oxide cluster that the Hupp group at Northwestern University has demonstrated in metal organic frameworks (MOFs) is quite active. However, you cannot inject MOFs

into people. They are crystals. So a soluble form that is 3 or 4 kilodaltons in size would be very, very useful.

Audience: This is a beautifully designed system for your molecules. It seems like it will stand up to a huge range of systems. How easy is that? How much work are we talking about to take the platform that you've built here and extend it, say, to designing nanomachines of Stoddart's type?

Christian Schafmeister: Stoddart's type is easy. With the new user interface that we are going to get working in the next couple of weeks, you would just sketch it like you would in ChemDraw, and then build the 3D structure of this, then this part, put the ring over the thread, and you will have a molecular model that you can then run a dynamics simulation on. Then you can run LAMMPS on it without writing it out to a file, shutting it down, and starting up another program. So that is easy.

Designing things like peptides, or peptoids, or other more flexible, floppy molecules is hard because they have these huge conformational space that the molecules can access. I think it would be great for setting up simulations, but I design these molecules this way so they will be easier to design because they will have well-defined shapes that I can predict.

I am working on a project right now to get atomically precise models of a DNA origami structure. That is something where you could get a defined shape, but it will have 49 million atoms in it. This platform (CANDO) can do scale. I have loaded 10 million atom structures into it. No problem, because the memory management, all of that hard stuff, is done properly right from the ground up. You can build and manipulate very large structures. Why I have been going off on Python all weekend, there is a program called ParmEd that translates input files for different molecular dynamics packages. It is written in Python. It is great; lots of people love it. But it doesn't scale. You cannot put more than 10 thousand to 100 thousand atoms into it, or it takes several days to load the file, process it, and write it back out again, because it is written in Python. The stuff I have written in C++, LLVM IR, native code everything, linked together, inlined together, so it runs fast, scales, and you can handle tens of millions of atoms.

Audience: This is a very exciting project. It makes a lot of good sense. You need to tie in the quantum mechanics. You can do that with VASP for up to 300 atoms [Vienna Ab initio Simulation Package "a computer program for atomic scale materials modeling, e.g. electronic structure calculations and quantum-mechanical molecular dynamics, from first principles" https://www.vasp.at/]. For larger systems You might think about using reactive force fields [to use quantum mechanics for large-scale systems, see page 14], which scale up to a few million atoms.

Christian Schafmeister: I want to do that, but I don't want to implement the force field again. I want to use your code; that's why I asked about it.

Audience: It is in LAMMPS already so you can start using it.

Christian Schafmeister: And LAMMPS is a C++ program, so we can take the C++ code, CANDO has built within it the ability to read C++ code using the Clang compiler and automatically build an interface to it, so I could get LAMMPS into CANDO in a couple of days.

Audience: There are a lot of codes implemented in it, and they all have different hairstyles and different details, but the point is that you can check it out with quantum mechanics and even with 50 million atoms and make sure there are no glitches.

Ben Goertzel: From an AI view, OpenCog is C++, the same AI algorithms can be used for the kind of molecules you are working with and for more atomically precise, Stoddart-style molecular machines, and so forth.

Audience: And AI can also help improve the force fields.

Ben Goertzel: Also true. As an AI developer, to interest other AI developers in trying out the tools for nanoscale design, it is important to have as simple and unified a platform as possible. AI developers don't mind dealing with LISP or various kinds of software, but they don't want to have to understand the details of different simulation packages, so having this uniform interface to different types of nanoscale design problems is key to getting the AI community engaged in this kind of work.

Christian Schafmeister: Chemists are going to be a little resistant to learning Common LISP, but there is a Python implementation in Common LISP so we can just compile into there so you will have Python, but you will be able to access Python and all the C++ seamlessly and have it all. It is all C++, LLVM. I think it is going to be a really great molecular design, molecular programming environment.

Audience: These are prophylactically injected structures in solution for soldiers for the purpose of detoxifying objects, but are we sure that the structures themselves, while they are floating around, won't bind with something that creates toxins?

Christian Schafmeister: Because these are rigid structures, they have groups that stick outwards and groups that stick inwards. We can change the functional groups on the outside to change the solubility and what they might bump into. We don't have to worry about how it's going to change the inside, because everything is rigid, stiff. If it does stick to something, we will just change the outside so it doesn't stick to that thing. But we can preserve the inside. We can resurface these molecules without affecting the internal structure. All of these structures are just made from carbon, hydrogen, oxygen, nitrogen. CHON is the stuff we are made from; it is safe, non-toxic. The solubility of our molecules is excellent. We purify them in water and acetonitrile. Because they are stiff, weirdly shaped, and point groups out in different directions, they are just not compatible with themselves or anything else. So they stay in solution. You have to design them to bind specific things. You could remove them from the body by making something complementary to pull them out. Also, they are small. These catalysts would only be four or five kilodaltons, and anything less than fifty kilodaltons will be cleared by the kidneys. They will have great properties for use in the body. We have thrown them at cells, and they do not hurt them at all.

Audience: Having worked in biotech, I know it is a long way from "we can put it in the body" to "it's going to work". A lot of things happen. What does the DOD see as the process of development? I am assuming that you are at an early stage in the development of a long-term project.

Christian Schafmeister: At this stage we are not really worried about getting it into the body. The molecules themselves have inherently good pharmacological properties. We have not looked at that in detail. We first need to demonstrate catalysis.

With nerve agents, you are going to die. They are developing enzymes that could protect, and they do have a

prophylactic effect in guinea pigs. The enzymes they have are quite active. They will have an immune response; they will have all sorts of problems. They are more focused right now on just detoxifying things. We are developing applications where we do not have to go through clinical trials right away. I would like to discover some molecules that bind proteins very tightly. We have one. If it looks like a good drug candidate, then I would definitely like to go and explore the pharmacological properties of these molecules.

Audience: You talked about that you could build channels. I think interesting applications of that are channels into organelles, as an example into bacteria, or even into plants to increase gas exchange. etc. On a basic science level, you have interesting systems that could be put together with biological systems.

Christian Schafmeister: Yes, that is why we make them thick enough to span a lipid bilayer. We could put greasy groups on the outside, and I think the structures would go right into the membrane and hold open a hole.

Audience: Could you orient them in a specified direction to get selective transport?

Christian Schafmeister: Yes, because we can add groups to achieve selective binding. They are asymmetric, the top is different from the bottom, so there should be a selectivity of how they go into the membrane.

Audience: Could you regulate the direction that molecules go through the channel to get one-way transport?

Christian Schafmeister: With more complex structures having moving groups inside that can close and open to create selective channels, and even pumps are possible. Proteins do everything like this. With big enough structures I am sure we could do that. It is going to take a lot of engineering and design.

Audience: To you have the groundwork to scale up to that?

Christian Schafmeister: Yes, the synthesis of these are all convergent. We synthesize segments, and then we link them together. We try to keep the number of linear steps short, because you lose every time we do the chemistry. I think we can assemble large molecules that are five to ten kilodaltons in a fairly straightforward fashion. Beyond that would require more synthesis. For membranes, this stuff really scales. Five grams of this would cover a football field of membrane. For active catalysts, a small amount can generate a lot of product.

What Software and/or AI Does the Nanotechnologist Need?

Design Molecules, Explore Functions, Screen Designs Quickly

Audience: I think to move in this direction you have to have software that is modular, and if you want to design molecules you need software that will build molecules and that will explore different functions of the molecules and will search through millions of designs quickly.

Audience: You have a lot of expert knowledge about what properties that molecule should have in order to have a good score, which can then be placed like fuzzy or soft constraints in the fitness function for advanced algorithms.

Audience: Right. My software exists, the force field, the pieces are all there, but integrating it together so that I can try one problem today and another problem next week, that doesn't exist.

Audience: That is sort of what I was thinking at the end of our last workshop. In robotics, we have Gazebo, a robot simulator with models for each motor, models for each gearbox, etc.. If you want to use AI algorithms to simulate a design for a walking robot, it is not trivial, but you can piece together what kind of robot you want, and start with any AI algorithm and run the simulator. I think something vaguely like that in this domain would make life a lot easier.

Audience: That is what I have been working toward; that is what I would like.

Audience: The kind of software should be able to identify the best 100 sites, but the quality of the force field, the way in through the solvent, may not be able to tell you which of those 100 sites is the best.

Audience: My main concern is to stick with this thing. I was trying to write this 5 to 10 years ago. There is basically a lot of C++ code, when you get down in the weeds, and you have to get down there because we are working directly with the processors real time, so you have to write a lot of C++ code, then how do you tie that together? We are trying to write high level chemistry software that knows Prelog rules <https://en.wikipedia.org/wiki/Prelog_strain> and builds 3 dimensional chemical structures, but you got to do that in a more dynamic language where you can do programming. Hooking these kinds of modules into Python turns out to be very convenient, but Python is very slow. I'm involved in a programming environment which uses a compile language to tie together and interoperate C++ and Fortran so I can take C++ and Fortran libraries that evaluate force fields, and tie that in with code that builds molecules, in a very tight loop that does not involve Python.

Audience: But Python doesn't have to be … I mean Tensor Flow …

Audience: It turns out it is, because if you try to do really complex things with it, you start to see more and more work, and it is an interpreted language, and it just becomes very problematic.

Audience: Right this minute, I would like software for something that is about 3000 to 5000 atoms. Something that does modeling and design. More is better. I want the AI people to define for me what molecular parameters they would need to train the systems. For example, I can go and take a lot of molecules and get spectroscopy data. Does that help them? I can go get X-ray crystallography data. Does that help them? What do they need in order to train their systems?

Build a Target Structure from a Sequence of Monomers

Audience: I've got a slightly different sequence in that I see two basic processes that I need to solve. One is a sequence of monomers that builds a structure, and what is that structure going to be? That includes issues of bond

angles and minimization of conformational energy and optimization of pi energy systems. These are all aromatic monomers and polymers. But that might be an entirely different type of challenge from the structure-function determination, which might lend itself to an AI application. But I'm not looking at that as a big challenge because the idea is that this is a design environment and in two hours I can create a new prototype. What I want to do is create different prototypes and qualify them experimentally and functionally. I need to see how these structures are going to perform once they are assembled. If they are going to be polymers with metal atoms in them, then where are the metal atoms going to be? What kind of proximities are there going to be? I am more interested in that first step before we get to even looking at functionality.

Editor's note: For more on the "sequence of monomers that builds a structure", see Atomic Precision for Energy 2016 pp. 11-19, especially Fig. 4.

Molecules on Functional Scaffolds or Surfaces

Audience: I have a third type of question since me and my colleagues are likely to be the ones to actually try to make it work from an engineering point of view. Analyzing a molecule is not enough. We need to assemble it on a functional scaffold. We can assemble it using DNA origami, we can assemble it using scanning probes, we can perform a lot of interesting manipulations using electron beams. For many of these techniques we can use partial quantum effect libraries. For example, we can shine an electron beam on this part of the molecule, or we can apply a voltage to this particular part of the molecule to initiate some particular pathway. The question is can we somehow combine machinery to learn from experimental examples, and develop the theoretical tools that allow us to predict how molecules change under the action of this specific kind of manipulation. We should be able to figure out how to assemble them, how to edit them, and so forth. Any molecule. For example, I have a naphthalene on a graphene sheet; I shine an electron beam on an atom or 10 picometers away from an atom, I want to figure out what trajectory will have what effect. On the one hand, this is a very complex problem; on the other hand, this is something that can be done experimentally. I can go in the lab and I can do this. I just have no idea what will be the result.

Audience: How does a chemical reaction translate to a specific surface? We need to know from an experimental point of view what can a simulation tell us about how we can work with a molecule on a surface so that a specific chemical reaction can occur.

Maintaining and Updating Current Software

Audience: There was a lot of talk earlier about integration. There are a lot of problems that I wished software people solved, and that I am trying to solve too. There are lots of tools that can do various materials scale, molecular scale predictions, but some of them are experiencing software rot as the product was introduced 15 years ago and has not been updated since. There are tools that you would like to tie together. I think we do need and don't really have a good tool for designing supramolecular systems. AI would be really great, but I think there are a lot of mundane, super-low-hanging fruit, like user interfaces, better documentation, writing a modern operating system, that are huge problems right now. Not the sexiest problem to solve, but they need to be solved.

Enzyme Suite to Turn Carbon Dioxide into Diamond in Sunlight

Audience: A trillion dollar bounty on a suite of enzymes that turn carbon dioxide into diamond in sunlight. It would revolutionize materials science. N-crosslink breakers for advanced glycation end products. Very focused. It would save a lot of lives potentially, or at least test some hypotheses about the importance of extracellular crosslinks in aging. Also, enzymes to get rid of junk that builds up inside of cells. We need enzymes for Aubrey de Grey's whole SENS platform.

When You Have a Lot of Data But No Clue What Is Going On

Audience: We have been working on nanotechnology at the molecular level for a long time, and we have been doing it the only way that we can do it: using models of physics, using models of chemistry, using models of mechanics at the atomic level. The kind of AI that I do is dealing with systems where you do not know what the model is. My wish list is basically what problems do you see in your work that you cannot figure out the model, but you have a lot of data. That is where you use the second half of the toolkit that I talked about earlier. What problems do you have where you have no clue what is going on? But you have a lot data and you want to figure out the model.

Database for Molecular Machines

Audience: A database for molecular machines. The problem with most of the studies of molecular machines is that structures and 3D coordinates of the molecules are not well-validated, so it is really difficult and time-consuming to draw all of these things by hand to study them. If there is a tool that just by looking at the description of this molecule could generate a structure, then we could study these systems in detail and study the motions of these machines in a simpler way. We could even use these databases to build bigger databases to understand how things operate at this scale. We could use such databases to build systems of molecular machines.

Binding and Pathway Data to Unify Cell and Protein Structure Data

Audience: I am a molecule/software person with molecule/software issues. In biology we are going from the top down and we are seeing increasingly large databases of protein localization data within cells, and from the bottom up we have X-ray crystallographic data giving us atomic coordinates within crystal structures, but what I would like to see is more unification of the two. I would like to see people figure out what protein complexes might look like based on binding data or pathway data.

Relationships Among Molecular Structures, Roles, and Applications

Audience: I am not a chemist, but one thing I would really like to see is, if you look at all of the numbers and all of the scientific data that have been generated, what are the facts that some molecules have different roles, what are the cause and effect relationships, how can we extract that in a way that we can actually plug into an application, and then how can we best model a universal view of what is happening.

Protein Design & Ligand Binding: How Much Data Is Enough?

Audience: My question for the AI people is how much data do we need to describe the space, and not simply overfit it?

Audience: It depends. The best thing you can do if you are architecting these efforts is create new parameters and see how much it overcooks, because some software models have much fewer parameters and need much less data. Another trick worth looking at is where you take a dataset and its image, for example, and rotate it, skew it, or make it blurry to artificially increase the number of examples, but it is something you have to experiment with. I got the impression you and your colleague David Baker had some really great protein design results where you could specify a shape, and you could design a protein that would fit that shape to within a few nanometers. When you are starting with an amino acid sequence, you could get that shape, or pretty close to it. Right?

Audience: Yes, but there are many other proteins that are especially hard to get data for, or have not been expressed. For example, the problem of ligand binding. We can get more data, but how do we put this together?

Audience: There was a guy from Google here last time [AI for Scientific Progress Workshop, Sept. 30- Oct. 2, 2016], and he told me they had 125 billion binders for a single protein, based on DNA-coded computational chemistry. Wouldn't that be very useful, to have a single well-defined target and 125 billion variants that bind to it.

Audience: But not necessarily to the same site on the protein. A crystal structure gives you all the residues and the structure and where the ligand binds, but for each ligand the residues will have different conformations. The crystal structure does not tell you the conformations of the residues that interact with the ligand to stabilize the binding. We have software that takes a pharmacophore and goes through a couple million molecules to find the thousand or so that are best. It can narrow the field from two million to a thousand molecules. In those thousand you will have the ten or twenty good binders, but that software will not tell you which of those thousand are the good ones because it cannot identify conformations that match the pharmacophore. The problem is going from those thousand down to the one or two that might be nanomolar binding. Right now, the quality of the force field, and the issue of handling how the residue side chain conformations change—these are things that people know how to do, but it is not systematized well enough that you could trust a program to give you the right answer. Even the quality of the force field is not sufficient to give you the right answer.

Genomics: How Much Data Is Enough?

Audience: I have done a lot more stuff with machine learning applied to genomics. The answer there is that it depends, but I have a better sense there of what it depends on. How much data you need depends on the combination of genes and the complexity of the phenotype. You have the phenotype you are looking at; you have 14,000 genes in a fly; 35,000 in people and all of this epigenetic data, so if your phenotype fundamentally depends on the combination of a hundred different genes ...

Audience: If you have ten or twenty genes, how much data does this mean?

Audience: For SNPs (single-nucleotide polymorphisms) or gene expression data, if you have a relatively simple phenotype you could have dozens of cases and dozens of controls, even though you have 25,000 genes characterizing each one. If you are looking at something like Alzheimer's disease or Parkinson's disease, what we work with in practice is something like 500 to 2000 cases and an equal number of controls, and you can discover something. You have about 25,000 genes in each person. Now how much better data you would get if you had a million cases and a million controls, we will find out in the next few years. At the moment, we don't know. We are using an estimation of distribution algorithm over the program free space. It is probabilistic evolutionary programmed learning. There is a lot of tuning to the Occam's razor bias, which is a component of the fitness function. It is a particular challenge to learn models that do not overfit the data. You set aside about three quarters of the data, then you do cross validation and so forth. It is blackbox in terms of the models, but you are assuming parameters on subsets of the data.

Developing AI Proposals for Designing with Atomic Precision

Approaches to Designing Large Molecules

Audience: In computer Science and AI we have a problem that we have been working on for decades which is isomorphic in many ways to the problem of constructing molecules, and that is the job of constructing a sentence that makes sense. We have gone through three generations of technology, for instance, for how to make machine translation.

- We have done it using rule-based systems, using grammars, and using models. We used those for decades.
- And then there are genetic algorithm approaches to exactly the same problem. We have assembled a population of sentences, evaluate them for their grammatical correctness, take the winners, cut them in half and breed them with each other, and you get better sentences over time. This is basically how these things work. So this is the second generation.
- The third generation is deep learning, where we have the system learn a representation of the sentence from the training example, and then it basically generates another representation.

So if Deep Learning can scratch that problem of machine translation, I think there is a fair chance that deep learning can scratch the problem of constructing useful molecules also. You can just transfer the algorithms from one domain into another.

Audience: What forms the input to any kind of learning model? For many problems there are no degrees of freedom and no well-defined classifiers. So it would be great if when thinking about a specific architecture, you also think about what goes inside.

Audience: What I was proposing, first of all, was a software framework that glued together various components in an easy to use way, so that it would be a wrapper where one could easily plug in different simulators, and could easily plug in different AI algorithms that serve different functions, such as design or accelerate a simulator. Now within that framework, my instinct was actually to try evolutionary learning first for the design problem, and deep neural

networks first for the accelerate the simulator problem, so the specific solutions that have been tossed around here are essentially the same ones that I suggested to try first. But I think there are two aspects here. One is having a software architecture that makes it easy to experiment with different simulators and with different algorithms in different places, and with whatever AI algorithms you plug in there. We have been discussing the AI algorithms, and the first guesses of what to try have been much in line with what I've described.

I've gone a couple steps further in some ways. Evolutionary learning is a broad domain. One thing I have been working on is putting evolutionary learning and probabilistic logic together so you can use evolutionary learning to learn a small computer program, which is genetic programming such as John Koza pioneered. But then you could also do logical rewriting of these programs using inference rules, and you can do inductive or abductive reasoning on these programs to get new programs that can then be tried within the Genetic Algorithm population so that I'd been thinking to try evolutionary learning combined with some probabilistic inference to generalize the evolved models, so in a way you could do that as a refinement of the idea of using genetic programing, or you do it as a different AI approach. I think there is going to be a lot of experimentation we want to do within different AI techniques here. So having a good robust easy to use platform for experimenting with different methods is as important as choosing the right initial method.

One could also use genetic algorithms to learn to structure deep neural networks. There are a lot of variations one can think to do.

AI Framework to Address NanoDesign Challenges

Facilitator: Let's look at each of the needs that the people who work with molecules have just identified, and see if we have software people who want to propose an approach, whether it is an AI approach or a more traditional approach.

Additional Challenge #1. Predict molecule behavior during assembly; connect to experiment; in the environment (substrate or liquid); action of local stimulus, probe or e-beam mediated assembly, editing, or modification.

Audience: It is addressable. This is fundamental physics on the interaction on surfaces of light with electrons. It is not easy; it might work for the electron spin, but I am not sure. At one level, looking at how molecules organize on surfaces is useful for lots of things. Again, it's a pretty broad topic. There are a lot of different molecules and surfaces: metals or others. That's the kind of stuff that we should be doing right now.

Audience: From an experimental point of view, it is possible to collect the library of coordinates following an action. The question is, if you have this kind of information, is it possible to reconstruct what is inside the black box.

Audience: It could be an important feature to test out new algorithms on something that we already know how to do.

Additional Challenge #2. Integrate and Maintain software tools, address aspects (e.g. scales) of molecular machine design, stack of simulation tools with speed / accuracy trade-off, wrapper for applying various AI methods.

Audience: To provide easy access to a diverse combination of simulation tools is a software design problem more so than an AI problem. In robotics, for example, a big step forward in academic robotics was the release of ROS (Robot Operating System) by Willow Garage. ROS is essentially a messaging platform and protocol. You have the computer vision algorithm, the arm controller, a planner, etc., all wrapped in an ROS code that takes in and sends out messages. Since everyone is using ROS, people know what that means, so they can assemble a robot control configuration for various ROS nodes that may be written in different languages or using different paradigms. It is easy to put these things together. Something vaguely of that nature could take simulators and simulate different types or aspects of physical systems, and then in some standardized way it gives them information and gets information from them without have to figure out how each one of them works.

Audience: People use REST (Representational State Transfer) for designing large systems and separating components from each other. It is default architecture that computers use to send messages to each other. That may be useful for our purposes, but REST is a pretty low level protocol. We may need something that is higher level, or we might need a layer on top of REST. Another useful tool is Docker. If you have something complex to install or configure, you can supply it in a Docker container. That solves a different part of the problem. The take-home message is that we have solutions to solve really large problems in clouds using standard tools that have existed for a decade now.

Additional Challenge #3. Design biomolecules and catalysts. Examples: CO2 + sunlight --> Diamond; crosslink breakers; better CRISPR gene editing; Coarse grained protein complex inference

Additional Challenge #4. Database for molecular machines; tool for image of molecule, atomic coordinates, bonds; apply machine learning to these databases; study a number of systems in detail to understand which kinds of simplifications can be made to study molecular machines more efficiently; data augmentation

Audience: What software technologies are needed to address this problem? One question that comes to mind is knowledge representation. What is the right way to represent these examples so that various AI algorithms can make sense of them. Would you describe a system built of carbon atoms in precise positions the same way as you would describe a system built from DNA origami, for instance. What is the right description language to use? How does describing molecular machines differ from describing large scale machines?

Audience: I've worked with large scale machines, and describing them formally is also difficult. For Hanson Robots, the motors and the strains are easy, but describing the face is challenging. It's a flexible material made by mixing organic and inorganic components, to which anchors from the motors connect to certain points inside the face. There are no good soft body models for the face inside the robot simulator. Exactly how you describe the shape of the anchors inside the face is not very clear. You have to improvise that. The rigid stuff is easy, but there is no standardization for the flexible stuff in the robotics world. That problem is what I wonder about in the nano world. If you have floppy organic molecules, what is the best compact way to describe the system?

Audience: Here is what makes it a hard problem. You can describe a molecule as a 3-dimensional structure with x, y, z coordinates for the atoms, but the properties that that molecule has can be determined from pain-staking experiment with some uncertainty for the measurements obtained, but from one molecule to the next the properties will vary so it is not exactly clear what you will put into the database. The key would be the 3D structure of the

molecule, but even that is not correct because instead of one static structure, the molecule is really moving a lot. Additional variables like what solvent is it dissolved in and at what temperature also contribute to making it a hard problem.

Audience: I am reminded of a similar discussion from algorithmic finance, where they wanted to use all this input from real estate and macroeconomic data, but they had a lot of partial data, with various pieces missing. So they had a lot of partial data, but it was data nevertheless. Molecular machines might be similar in that you have missing properties for some molecular machines, but other missing properties for other molecular machines.

Audience: These are quite complex problems. There are many things going on at the same time, and they all affect each other. But this is not new. The problems cannot be worked around until you have a really large amount of data. AI cannot do anything about all those interactions unless it has enough data to figure those things out. Modeling is the only way around not having enough data. Keep on working; we can't use AI until we have plenty of data.

Audience: How much can we constrain the system that we are looking at? Can we structure this so that we have a simulator, a structure-function analysis generator thing, plus the constraints of the systems that we are looking at. If, for example, you are looking at Christian Schafmeister's spiroligomers, you can constrain your simulator to that system and reduce the amount of training data you need. However, if you constrain your simulator to proteins you have very different constraints. If your system is constrained to all chemical space, you require an enormous amount of data. If you constrain your functional search to a limited chemical space, how much data do you need?

Audience: I think what I am proposing is that we start with a small protein domain and try to use those methods for the grander molecular machine design problem. But we have to have enough data.

Audience: As an example from our work, if you are designing a pore for a membrane that you want to be specific for a certain solute, you have to do 'methyl-ethyl' chemistry; i.e., you make small changes that slightly change the overall structure of your building block. If you don't take into account the ability to do small molecular changes, you will not be able to do a big design. That is why the idea of a fixed building block is difficult. However, a fixed building block with the ability to make small changes in almost any atom to give you the properties you want, works fine. But you only get out of it, what you put into it. If all that goes into it are amino acids, all you will get out is protein.

Additional Challenge #5. The scientific literature: computationally useful info on molecules. (1) extract cause and effect relationships in different environmental conditions; (2) create universal cause and relationship map/model

Audience: this is a more general formulation of the idea that we have been looking at. The idea is really that there is already in the literature a lot of information that has been generated in useful formats. The question is how we extract that information from a jumble of figures and text, then inspect the cause and effect relationships, and then we create a universal map that becomes better leveraged than all the stuff that is already out there. More than just structures and properties, this is about dynamics and kinetics of transformations.To get to the deeper questions that we are talking about, we have to have a deeper understanding of the text. It is not enough to just find the keywords; that is what we did 20 years ago.

Audience: It's not clear that it is harder. It's just a different type of problem. I am working now on unsupervised

learning with large corpora of datasets with some interesting success. So that's a problem the AI community is focusing on more, with a lot of resources, from different directions. Google and Microsoft and other companies have their own multiple approaches to this problem, which they are working hard on now.

Presenting the NanoMind Proposal — Ben Goertzel

Ben Goertzel: Among the working groups we had last time [Artificial Intelligence for Scientific Progress, Sep. 30-Oct. 2, 2016, overview video, workshop white paper], one of them was focusing on AI for nanoscale design, and actually there was another one that was focused on using machine learning to accelerate quantum chemistry simulations, so what I am going to discuss now is a mashup of those two things, plus some discussion of software integration mostly inspired by the last workshop, but with some heavy post-processing by me afterwards, which was done in conversation with various nanotech people.

One element is simulators. It is clear from our discussions that there is not one monolithic nanosystems simulator; rather we have a host of different simulation software to simulate different types of nanosystems. Making a way to interface these various simulation packages is one problem that we face.

We also have the domain of simulator simulators: using a neural net to simulate what the actual simulator is producing.

And then we have a learning algorithm that will look at the actual simulator and figure out how to make a simulator simulator. Deep neural nets are one promising candidate for that function. People have used deep neural nets to simulate physics engines, which gives some promise for doing this in this domain, although there are certain to be additional complexities here. Then we have AI design algorithms. That is where you could have an evolutionary learning algorithm, a logic engine, a combination of those things. These algorithms need to use the simulator for evaluating candidate solutions, and potentially also for gathering information.

A human being is going to specify what algorithms they want, and that is another question. Something else that has come up in discussions already today, is that there is some general know-how about the various domains involved, which is useful for guiding the AI design system. As someone was saying, when I design a molecule, I want to make it floppy, but not too floppy. This kind of domain-specific know-how will be important guidance to the AI system. What is the right way to formalize that sort of fuzzy, soft know-how to feed into your AI system is another problem.

Another aspect of all this is data. Some data will come from actual experiments. Some will come from real or simulated simulations. Another big use for AI will be to analyze all of the data that comes from this. In fact, some of the general know-how could also come from AI. It could extract know-how from literature using NLP (Natural Language Processing). If we had a true AGI (artificial general intelligence) system, then all of the diverse functions assigned here to AI would just be different cognitive functions of same unified AGI system.

In this very high level picture there are a lot of questions that have to be addressed. Sometimes the most fun questions to think about are not the most fun questions in terms of building a successful system. The most fun questions for me to think about as a mathematician and AI guy are what are the fundamental algorithms and

representations that we want to use here. How do you actually use deep neural networks as simulators for quantum chemistry or DFT (density functional theory)? This is much higher dimensional than two dimensional images. Are you doing convolutions over Hilbert space, for example? There is a lot of interesting thinking about how you would architect a deep neural net to emulate these simulators. You could have a neural net with lower layers of feature detectors specific to your domain. The higher levels could be more generic pattern recognizers. We could also plan with probabilistic programming as a way to learn deep neural networks.

On the AI design side, genetic programming I think is the right very high level paradigm, but then you start thinking about details. Say, you are using an estimation of distribution algorithm (EDA). In classic genetic programming you have program trees, which in the simplest case are trees of LISP functions, and you evaluate the program trees on doing something, like making a nanodesign that the simulator says fulfills your specifications, then you can mutate and cross over those program trees to create new program trees that you then evaluate. A program tree could be interpreted, for example, as a series of instructions for manipulating something that builds the system. In an estimation of distribution algorithm approach, you have these little program trees, each one of which is a program that controls a manipulator that builds the system, but instead of doing only crossover and mutation on the population of program trees, you build a probabilistic model that studies the successful program trees versus the unsuccessful program trees. From that probabilistic model you then do instance generation to get new program trees that might be successful at moving your manipulators to build your system.

Now how do you estimate that distribution? You could use a neural net to do that; you could use Bayesian inference to do that. In this case, if your program is building a quantum system, do you want to do a quantum logic inference to extrapolate your program tree status? There are a lot of fun things to think about in the details of that AI algorithm, and in the details of the algorithm for simulator simulation, etc, and also in data analysis, but I am focusing less on that because it is so well understood because so many people are analyzing complex data. Those issues are the fun ones from a computer science and AI point of view.

On the other hand, there are going to be a lot of people experimenting with a lot of different things here. I think probably the more critical hurdle to overcome is to somehow make an easy to use framework that wraps up all this stuff. If you have such a framework, then a lot of smart people can try out a lot of algorithms. If it is a pain to connect your AI algorithm to your simulators and the data, etc., no one is going to do it. If you look at what has driven the application of deep neural nets and all these other algorithms to image processing, part of it is that we have large corpora of images, of faces, of natural scenes, etc. Any one can download the database of images of faces, of cups, of cars, etc.; there is an easy evaluation function (how many did I get right); that makes it very easy to run these experiments, which has pushed progress forward tremendously.

Part of the problem with robotics is the lack of such a framework. Your framework is a robot, and if you are an AI guy, maintaining the robot is a pain; 99 % of your time is spent keeping the robot from breaking, replacing motors, connecting loose wires, etc. Here we need to get a framework to be relatively easy to use so that people whose

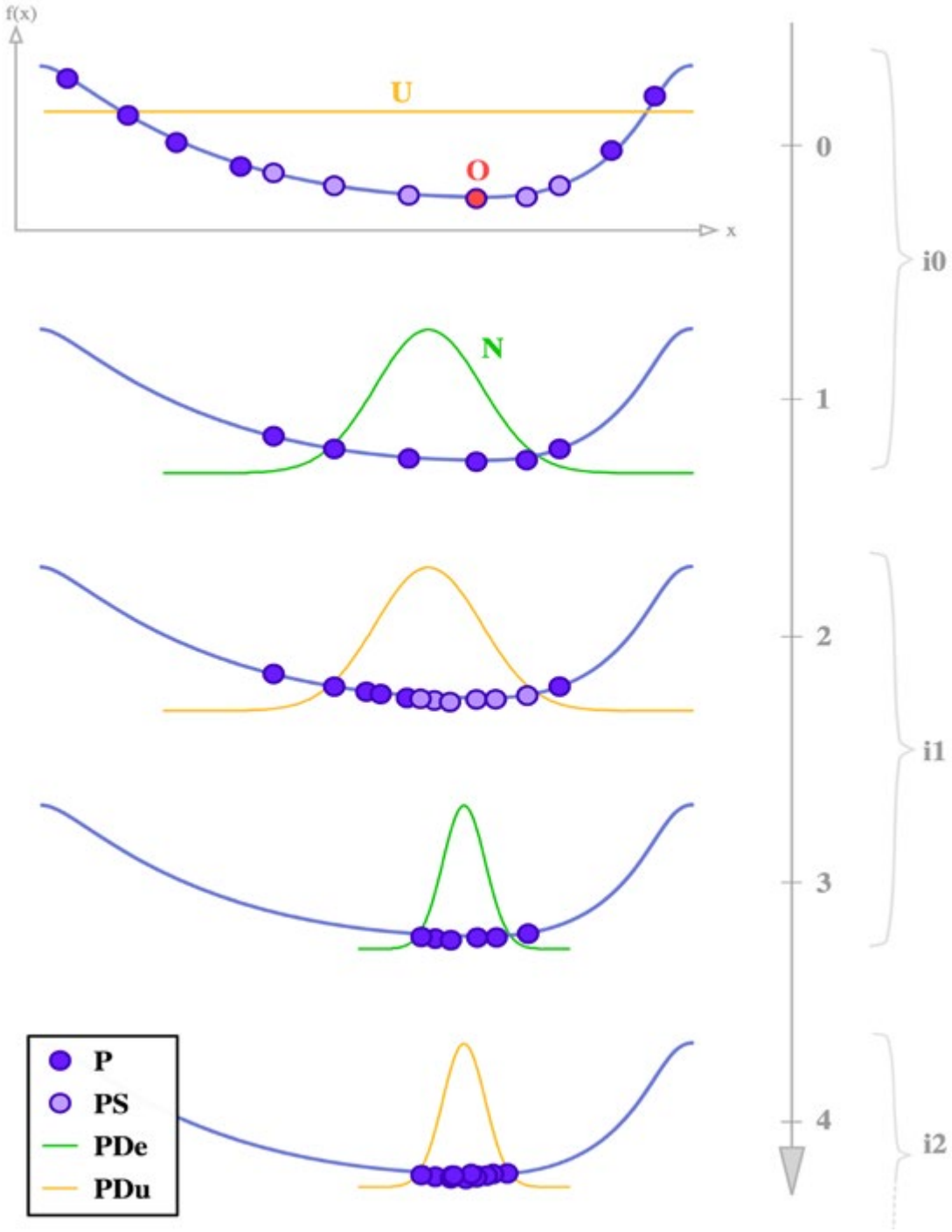


Figure 37. *Estimation of Distribution Algorithm (EDA). "Estimation of distribution algorithm. For each iteration i, a random draw is performed for a population P in a distribution PDu. The distribution parameters PDe are then estimated using the selected points PS. The illustrated example optimizes a continuous objective function f(X) with a unique optimum O.*
The sampling (following a normal distribution N) concentrates around the optimum as one goes along unwinding algorithm." https://en.wikipedia.org/wiki/Estimation_of_distribution_algorithm
https://commons.wikimedia.org/wiki/File:Eda_mono-variant_gauss_iterations.svg
This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported.
Author: Johann Dréo (User:Nojhan)

main interest is AI can try out AI algorithms on this type of problem. By the same token, the people whose interest is nanotech can easily connect a simulator of their choice, specify what constraints they have for the system they want to build, and what do they want the molecule(s) to do, and then just try a few AI algorithms on their problem. It has to be easier to do that than to solve the problem by hand or move on to another problem.

I think that architecting and implementing that framework is largely a matter of software design. The first step is requirements gathering from a diverse community, which we are doing the first 5 % of here. Then comes software architecture and implementation. That batch of work is why I thought it was appropriate to work with Foresight to develop a research proposal. Tweaking my AI algorithms to try to learn nanodesign is something I might do just for fun if I could find the spare time. Building this sort of infrastructure is not something I would do just for fun. It is a lot of tedious work, but it is very valuable once it is done.

Discussion

Audience: It seems to me that the simulator is the core here, to reference your analogy as to why the neural net field had a community of people who could easily try experiments and amass data. Your simulator of simulator assumes the simulator. The deep learner's learning of the simulator assumes the simulator.

Ben Goertzel: We have a lot of talent ...

Audience: No, but you don't have the equivalent of what Berkeley did with SPICE in the 1970s (Simulation Program with Integrated Circuit Emphasis). They came out with a really good simulator that covered electrical engineering and by inference all of the other fields, like mechanical engineering, etc., that revolved around the same set of differential and integral equations. All the AI design algorithms always assume a scoring function, which will always be a simulator. So what is needed is the Berkeley SPICE of nanotechnology.

Ben Goertzel: Maybe more diverse than that, though. I've looked at BigDFT (BigDFT website, https://en.wikipedia.org/wiki/BigDFT) which is cool and multiple machine distributed and uses GPUs (graphics processing units), but unfortunately that only addresses a certain class of nanotech problems. It's not clear to me that at the current state of technology you can make one uber-simulator for every nanosystem that everyone in this room cares about. So we are probably faced with a bunch of simulators instead of just one simulator. If that is correct, then we are faced with the problem of, as much as we can, making a common interface to a diversity of simulators.

Audience: What is the minimum information about an object you need to know to supply to the simulator?

Ben Goertzel: That would depend on the nanosystem. [addressing Audience] How many types of simulation software do we need to deal with? What is the commonality?

Audience: Molecular mechanics is handled pretty much by one set of functions. One set for DFT. There are many types of coarse-grained models. I see what you show on the board as a bunch of software elements that are connected to each other. What I am trying to develop is a framework where we can do exploratory programming that brings all of these things together.

Ben Goertzel: Do you think that the software framework we are developing could serve as the basis for all this?

Audience: Yes, I think so. That is our whole goal. I did not know that you guys came up with this last; I was not here last year. A tree-based software framework that would allow you to drop into it different computational components and they would all work together, and coded directly so that it runs fast.

Ben Goertzel: What Chris showed me a few minutes ago is something called Cando, which is LISP software wrapping up a C++ backend, which encompasses some aspects of what I've diagrammed. On a purely software level, it has an interesting overlap with what we are doing in our OpenCog system, because OpenCog is a C++ backend and mostly a Scheme front end, Scheme being a dialect of LISP. There happens to be a lot of software similarity there, which is not a deep point, but it is a practical point.

Audience: I think it also depends on what kind of a system you re looking at in terms of inputs and outputs. You may be looking at a catalyst where your primary inputs are hydration, temperature, and substrates, or you may be looking at a catalyst that has optical properties, where you have sets of frequencies in and catalytic properties coming out. You may be paying attention to entirely different sets of variables because in one case thermal energy is going in, and in the other case optical energy is going in. The whole idea of a modular system is ideal because you just use those modules that are related to whatever it is you are working on at the time.

Audience: Some of you have Fortran programs you would like to run as part of this simulator. One argument for using Docker gives you a box that serves as a virtual machine. You can put anything you want in that box, including a Fortran program. It allows you to use whatever you need at the moment.

Ben Goertzel: There are many layers to the architecture. We use Docker and AWS (Amazon Web Services), and then also other cloud providers to wrap up our Scheme and C++. There is an implementation layer where you want a container, and you need to define what protocol to use between containers, which may be REST. It is important to use standard software engineering practices to make your stuff distributable. Not everyone does in the research community. The next level up is what are the APIs (Application Programming Interface) for communicating between these things? Now, with different research software for different types of simulations, there is nothing approaching any standardization of APIs, and there are a great diversity of data formats. So I think a lot of the work is on the level of figuring out clean, comprehensible APIs for communication among the different components.

Audience: I just want to add that this stuff has to be fast. there is not enough processing power in the world to do a quantum mechanical molecular dynamics simulation of a standard protein; it doesn't even scale properly. If you go to lower resolution levels of molecular mechanics, it is an open ended problem to simulate enough molecules to properly predict properties of those molecules. We need speed! We need thousands and hundreds of thousands of processors running as efficiently as possible. Using solutions like Docker to send information back and forth will lead to huge communication bottlenecks ...

Ben Goertzel: Or with DFT, if you wrap it in an Nvidia Docker container, that lets you pass it directly to the GPU, where the work is done ...

Audience: My main point is that it has to be fast. I can integrate things so that they are running in the same machine in the same memory space, and doing as much as possible in one processor.

Facilitator: If you had access to such a tool would you use it?

Ben Goertzel: From my own standpoint, given my ignorance of the practicalities of doing nanotech, what would be useful to me would me to run through a few specific uses—cases of design problems that could be done with this type of framework. That would expose a lot more questions.

Facilitator: Considering Ben's NanoMind proposal and the proposals that resulted subsequently, could these all fit into one proposal or are there several separate proposals?

Audience: I see two levels for the simulator: one which would be structural in terms of looking at how all the pieces and atoms go together to make your structure; then there is the structure-function relationship as well. That one is dynamic and application-specific, and that's probably multiple simulation runs.

Audience: I think there is significant overlap. This idea of a chemical reaction, and what kind of a matrix is it in—in the bulk, on the surface, in a solution—is an overarching problem. With the Simulator I could look at the matrix it is in.

Audience: I fully agree that we just need to extend our range of simulation to the effective environment.

Ben Goertzel: I would like to take more than one specific use case and see how this high level NanoMind approach fleshes out in different cases. If, for example, we were to build a general simulator for robots, it would be important to see how it works in a walking robot, a rolling robot, and possibly also a flying robot. So here I would like to take some system where you are building with carbon atoms, like Legos; then some squishy DNA-type system, and then you need something else, which would involve different simulation engines, and different fuzzy constraints, and so forth. Then we could see how the high-level framework works out in those three specific cases. The high-level question with this sort of general framework is, are the different sub-cases you have to deal with too different for one general framework to make sense of them, or is it useful to make a general framework that abstracts from many different cases.

Another good thing to do is make it Open-source so others can contribute, but to make it even more Open-source than Google has made TensorFlow, which is sort of Open source but closed strategy. Google makes the TensorFlow source public after extensive development in house. I would like to develop this more Linux style, with open mailing lists so that people can constantly contribute patches and so forth. Being Open-source is a necessary but not sufficient condition for getting what we want.

Audience: How will this framework reference existing software or simulation tools?

Audience: You write an adapter for each one.

Ben Goertzel: If you forgot about using neural nets to simulate simulators, then the proposal would just be to make simple wrappers for existing simulators, which is what Chris Schafmeister has in his Cando framework. He

has LISP wrappers for simulation software, which provide a simple interface for the simulation software. One thing is just to wrap existing simulators in a common and standardized way so that it is easier to use them. Then another part of the suggestion is to use neural nets to simulate simulators to speed them up.

Audience: In Rosetta, for instance, they have tools to design protein-protein interfaces. So you have one protein for the surface, and you have another protein, and you want to design the surface of one of those proteins so it binds tightly to the other one. There they use Monte Carlo simulated annealing. I would like to go to the next step. There are a lot of chemists that make other kinds of molecules that are built up of building blocks, like beta-peptides and peptoids and spiroligomers and aromatic compounds. They can make structures that form regular helices. It would be really nice to use the same kind of techniques they use to control protein-protein interactions to control protein-unnatural molecule interactions. There are really good reasons to do this, because those unnatural molecules are not going to be recognized by the immune system; they are not going to be chewed up by proteases in the blood stream. But it is very difficult to get those unnatural molecules into Rosetta to run that kind of simulation. It would also be good to be able to take some of these neural nets and plug them into the software so that you could do some careful simulations of an unnatural molecule docking to a protein, and you could train a neural net to predict the binding interaction. I would like to build that into the system that builds the molecules and scores them. That is just one case study. Can we take what we are already doing with protein-protein interactions and transfer that to protein-unnatural building block interactions. We don't really have software that lets us do that in a simple, straight-forward way that we can put in the hands of the people making those molecules so they can then figure out what molecules they should be synthesizing to try and bind to that protein.

[NOTE: probably "unnatural" as it is used here would include both non-canonical amino acids and small molecules that are not amino acids.]

Audience: Would you be willing to say something about what is difficult about extending what is there to handle more artificial types of molecules?

Audience: Rosetta uses molecular mechanics; it also uses potentials that are cooked up from the Brookhaven Protein Data Bank. We don't have a Protein Data Bank for unnatural molecules, so you have to go back to first principles techniques like DFT, then parameterize the building blocks so you get all the partial charges on the atoms. I don't think Rosetta does any of that stuff. A chemist who makes those kind of molecules is not going to be able to take Rosetta and modify it to do those kinds of things. It is a huge research project

Ben Goertzel: If there are a substantial number of people who are interested in this high level direction, it might be interesting for people to divide into groups based on different specific use cases. One group could consider how this framework could apply to one specific design problem, and another group thinking about how this framework could apply to another specific design problem. To really understand this, I would have to understand how it applies to a few specific design problems, but I lack the domain knowledge.

Pitching AI for Atomically Precise Nanotechnology

The above wide-ranging discussions about what sorts of AI would be most useful for advancing atomically precise nanotechnology produced the NanoMind proposal for AI for Nano-Scale Design presented by Ben Goertzel and Christian Schafmeister. The discussions also brought out additional interesting perspectives on the development of Artificial Intelligence and the relationship between current AI progress and the eventual development of Artificial General Intelligence (AGI).

Technical Discussions

Ben Goertzel: What I've been looking at yesterday with Christian Schafmeister and John Koza was engineered enzymes in cavities using the special building blocks that Chris has produced. It seems if you could build these enzymes you could do an awful lot of amazing things. That's exactly the kind of application that I would love to see AI used for.

On the other hand, it's also nice to have somewhat more straightforward applications to experiment with a new methodology. What Martin was describing to me was some approaches in creating filters which would let certain substances through and not other substances through. We're looking at how you could design such filters using genetic programming, deep learning and so forth. The first relatively simple example that he described was to make a filter that would not let CO2 through. Basically he took different rings and offset them to each other so that the geometry of CO2 would not let it pass. A more complex filter he described combined benzene and cyclohexane rings with some extra complexity to allow self-assembly so that urea would not be able to pass but water and certain other things could. Discussion also included using single bonds vs double bonds to change pore size and stiffness to control what can go through and what cannot go through.

This is about the application to a particular type of filter, but it is cool that it has real world application to the developing world and the developed world, but from an AI point of view what we are interested in here is the relatively simple and tractable example where you can ask what is the substance I want to let through, what is the substance that I do not want to let through, what functional groups can I put on the sides to change the pore structure, and so forth. The task for the AI is thus to explore the kinds of structures with molecular dynamics simulations to see how well each kind serves in terms of letting the right things through and not letting the wrong things through, then you explore the data and generate more candidates.

There are additional details. To tune the molecular dynamics simulator they do some quantum mechanics simulations on a small number of cases, and use those results to tune the parameters. That kind of detail suggests to me that if you want to train a neural network to emulate what the molecular dynamics simulator is doing, one way is to train the neural network on the tuned molecular dynamics simulator after you have tuned the parameters using the quantum mechanics simulations. Otherwise you have a harder problem, which is to train a parametric neural network, which would operate on the same parameters that you tuned using the quantum mechanics simulator. That gets you down into the weeds. Do you have the neural network train a simulation of molecular dynamics in a quite specific sub-problem or do you try to train the neural network more generally that has free parameters that you adjust.

Overall, this is interesting as a not at all trivial but still relatively tractable test case for experimenting with these combinations of AI tools, genetic programming, estimation of distribution, and deep neural nets for emulating molecular dynamics. The thing you are building here is a static structure that has a certain function, which is great for testing, but ultimately you want to deal with dynamic systems also.

Another question for the group: If we are viewing Marty's filters as especially easy (and I still think it will be challenging to get the AI to do it) and then Christian's enzyme cavities are hard, but amazingly exciting, what would be an example of an intermediate in difficulty?

Audience: Almost all enzymes have coenzymes; you have an NADP+, NAD+, or something, and there the design part may be to replace some of those things with an electrode or something. Now you are replacing an element that makes it complicated with another element.

Ben Goertzel: Something else we were just talking around, there are generative adversarial networks, in which you have one network that generates structures, and another network that tries to distinguish the generated structures from real structures, and you co-evolve them so that then the generative network can generate the real structures that you like, and you can do that unsupervised or with supervision, and then you can add some latent variables to the generative network, the generative network then automatically evolves certain high-level control parameters in the latent variables. One could be "How big is your smile?", the other could be "How big are your eyes?" or "How long is your face" or "Do you have a beard?". So the network is automatically learning certain high-level semantic descriptors. What we are playing with now is running these networks with a Bayesian network of latent variables so the neural net, as it models the data, is learning a sort of hierarchical ontology of semantic descriptors, of features in the data. It will be quite interesting to see what those high-level descriptive parameters were when you were, for example, tuning the neural network to emulate a molecular dynamics simulator. What is the Bayes net of emergent control knobs for a molecular dynamics simulation. That is all a probability distribution, and then you can do various information theory measures of that distribution of the network of latent variables. We've never really tried that on data remotely resembling this before.

Chris Schafmeister: I think GA (genetic algorithm) and GP (genetic programming) for molecular design are immediately applicable. The neural nets to create a fast evaluator that is faster than a slow evaluator, especially in the area of molecular dynamics, it is really hard for me to see how that could work. In molecular dynamics, the slow step is that every atom has to see every other atom. You get a three dimensional arrangement of atoms that has to move and simulate the motion of molecules. I am not sure how you get a neural network in there to speed that up.

Ben Goertzel: What is the key difference between that and Newtonian or quantum mechanical simulations?
Chris Schafmeister: It is Newtonian.

Ben Goertzel: Neural nets can speed up Newtonian mechanical simulations. That's already been done in games.

Chris Schafmeister: So we have to simulate the motion of the atoms accurately, and to do that, every atom needs to feel every other atom's electrostatic potential and van der Waals potential. That is an N-squared calculation, and is kind of a slow step. Then you need to iterate that over and over and over again as you move the atoms.

Ben Goertzel: I guess conceptually what is happening is that when you have repeated configurations among many, many examples, the neural net is figuring out how to leap ahead in time rather than crunching through one step after another. It is guessing what is going to happen after many rounds of interaction in a repeated configuration.

Audience: They key word is 'guessing'. What we would like to end up with is a neural network that understands molecular chemistry at some level and makes educated guesses all the time, just like a human.

Ben Goertzel: If you have a bunch of billiard balls, and you note a familiar configuration that you have seen many times, you know that after a whole bunch of bouncing it is going to end up a certain way. That saves you from simulating all that bouncing over and over.

Chris Schafmeister: But if you take a garbage pail full of billiard balls, and you toss them down a hall, you can figure out where the balls will end up by simulating the motion down the hall. Where can you insert a neural net there to speed that up?

Ben Goertzel: In the midst of that dynamics, there are going to be many sub-problems that are very similar to those that have been seen in other cases of rolling billiard balls down the hall. When you identify one of these subproblems that you have seen before, you apply the neural net, and it leaps ahead to the result of many stages of simulation.

Audience: We are temporarily calling ourselves AttoTek. The idea is to combine AI/software tools with the atomic precision tools, which you will be hearing about, databases both public and private so that companies can work on proprietary projects but general users can also have access to general databases. The main functions are computation using AI to determine structures and structure/function relationships, functional groups of more complex systems, like molecules on surfaces. Some obvious examples include drug receptors, catalysis, electronic systems. We would like to see a maintained, open source system. There might be membership subscriptions to support maintenance after initial funding has been depleted.

Audience: The power of the tools is the rapid prototyping model that allows very efficient use of the system in terms of time and human resources. The building blocks are assembled into an atomically precise prototype, which is then screened, rated, and functionally assessed. You then go back to the software system to figure out if you need to tweak the building blocks to optimize the structure. Chris Schafmeister's presentation shows the building blocks being assembled into some kind of catalytic unit that you are modeling. Each one of these units is nicely predictable, and the crosslinks at the different ends allow the structure to fold together around a metal atom, and for all this to be modeled by the AI system. So we can look at a given stereocenter and ask about options for moving it, and each option can be modeled by the AI system. A successful application of this model has already been accomplished so we can test the model based on this system, and a variety of other systems that will be developed.

Audience: We want to extend this knowledge and predictions about molecular machines from the purely biological to the non-biological, to figure out how they work, and how to activate them, and provide the energy so they can do some function. A very deep part of the process is that we need not only to design molecular machines, but we need to find a way to integrate them into macroscopic applications. Our intuition is not going to be sufficient so we need to use machine learning to find alternative paths for integration.

So this is the theory, but the theory needs to be reinforced by experiment. On the experimental side, first we are limited by the number of papers we can read each day. Papers are published much faster than a human scientist can read them. The second part, is that if we have machines like scanning probe or electron microscopes, we ideally want to utilize all of the data that flows from these machines. During the ordinary process of analysis and publishing, more than 99 per cent of the data is never made available. We need to integrate experiment with theory so can capture all the data that is available so that we can see and make modifications on the molecular level. We need to integrate these processes of gathering and making available data so all of the data can be available for analysis by other interested research groups.

Audience: One problem is that different groups' data may not all be of equal quality.

Audience: Having the data available that is gathered by the microscope, but currently not analyzed, will help evaluate the quality of the results.

Audience: We are thinking of adding this unified software environment so that we can use all of the tools that have been developed so far. It is really hard to find out about all the tools, and learn about them, and use them properly. We want to integrate software packages that exist for chemistry into a unified molecular design package. The obvious benefit is that there are numerous software packages for molecular simulation; there are numerous file formats, there are legacy packages that are no longer supported; documentation is difficult or impossible to find; it is really difficult to go to the web sites to learn how to install and use the package. It would be useful to have a single user interface that we can use for all these packages. With common file formats, the output of one simulator can be interpreted as input for another simulation. All the software for file conversion and simulators exist, but we need to connect them to each other.

Chris Schafmeister: I think this is a really important goal. It is the central goal of what I have been trying to do on the technical side of things. I integrate a lot of software, a lot of other people's libraries to get CANDO to work. One of the things I did was to hook the front end of a C++ compiler into CANDO so that it can parse C++ code and analyze it, and build wrapper automatically for CANDO to talk to. I am going to apply this to other molecular modeling software so they can use those engines without having to re-write them. The ultimate vision here is to have CANDO read all that C++ code, build the interface to CANDO, so when the people who are developing other modeling software change a function, add an extra parameter to it, that change doesn't break everything. What it will do is re-write the wrapper, then check over the code that uses it to see if it is still using the old wrapper, and flag that so a programmer can go in and fix the interface. It is automated as much as possible, but there are some places where it is necessary for a human to make change. This might be a place where an AI could participate. But the automatic parsing of C++ code to automatically build wrapper is something that I am doing already. I am actually going with two user interfaces. Write your code in a text file, compile, and run. You can do OpenGL graphics and other libraries if you want a stand alone promo. But what I am going with now is client-server using Jupyter notebooks. So I hook a Jupyter notebook into my program, and you just bring up a browser and you can type commands into cells in a browser. It is kind of like Mathematica. We have a Jupyter package that is Common LISP support for Jupyter notebook. We are working to get graphical widgets working.

Audience: One of the user interface features of NanoEngineer-1 is that we were designing it to design mechanical

molecular machines. We had a lot of things that had nothing to do with molecules or atoms, like motors and dynamometers and widgets that you stuck into the structure, and you could program the amount of force that the motors put out as a function of time to see how the force was propagating through the machine. So the closer you get to doing mechanical engineering in the molecular field, the more you are going to want essentially a whole machine shop down in the software, and the inputs and outputs to that in your user interface.

Chris Schafmeister: To get this software all talking to each other, the LLVM library is the key technology. That lets you compile everything into an intermediate representation language and link it all together so it runs fast. The other compiler collection, the GNU compiler suite, would be really great, but it has walled off a lot of the internals so you cannot mess with it the way you can with LLVM. I am now inlining C++ generated code with common LISP code, and vice versa. That is what you need to do in order to get your program to run fast. You can't do this with an interpreter. You have to have compiled code that is linked together properly.

Monica Anderson: I have a couple of general observations that you might want to be aware of. The first one is that AI is basically intuition driven. It is understanding. These things are not fixed like in reductionist science. When you switch to neural networks, you lose the advantages of reductionism. You lose completeness and repeatability. You lose transparency of process, so you cannot see what is going on, and you cannot understand the results you get some of the time. All the methods that provide you with understanding (machines doing the Reduction) are losing all the advantages provided by reductionism (of scientists doing the Reduction, creating the Models and equations). You have to be aware of that and willing to make that trade-off. What you get instead is all kinds of things that humans are good at, but computers are not, such as abstraction, understanding, novelty, learning, etc. All of these things are basically not available in the reductionist scientific framework but can be attained using Holistic (Model Free, as in Machine Learning) Methods by learning how the world works, not by having it explained in equations (which is impossible except in limited domains) but by autonomously collecting lots of experience of what has happened in the past and matching current situations to past experiences in order to predict what might happen next.

[In response to a question on the timeline for advanced AI]: The problem I am working on is natural language understanding. I spent 17 years on that, and I have a pretty good understanding of exactly what we need. For work on computational principles you can expect a rather long timeline. However, deep learning has shown that these methods work, and provided you have a positive example that you can look at to start with, you can incorporate more and more from the example that works into your problem. I would expect a breakthrough in natural language understanding in four or five years. What would happen beyond that I cannot predict.

Other Approaches to AI for Atomic Precision

Discussion after Ben Goertzel's Introduction to Artificial Intelligence touched on the relationships among the NanoMind proposal, other software for atomically precise nanosystems design, and the eventual development of Artificial General Intelligence (AGI), which would presumably employ natural language understanding to deploy the entire corpus of human knowledge to atomically precise nanosystems design, among other applications.

Ben Goertzel: I think that as we work toward building AGI, having the proto-AGI solve important and interesting problems is going to be valuable for a lot of reasons. Historically most AI development has been military funded; now most AI development is funded with an objective of getting people to click on ads on the Internet. Each of these has

its purpose and role in society, but these AGIs, if we develop them to help in delivering medical care to people, and to help design nanosystems, probably we will get a better AGI that is useful to solve a variety of difficult problems.

Facilitator: I want to hear from the people who primarily come from the software field who have been working in this space, whether you call yourself AI or not. For example, NanoRex.

Audience: What happened was that a supporter of Foresight wanted to found a company to do something useful. He had this keen idea to build a piece of software to design larger things. It was going to be a highly targeted CAD system, and it was going to have all the physics built in so that you could put your atoms together and then simulate the thing to see if it worked, etc. There was not much of a market for that, so he was going to fund it out of his own pocket. Besides being one of the main, original architects of this piece of software, my other major accomplishment while with Nanorex was to get him to make it Open-source. So it is still out there. It may have undergone a fair amount of software rot between then and now, but it was a fairly extensive CAD suite. It was a large, highly featured, professionally written piece of design software for nanomachines. If you were going to put together a system that used AI to design molecular machines, you might start with this, strip out the user interface, and put in an interface to a high level piece of software that came up with designs. It was written mostly in Python.

I do have one interesting point that has to do with AI and design. Back when I was at Rutgers, we did a whole bunch of work with AI and design, mostly with the Department of Defense. A large defense contractor gave us some of their high-end software for analyzing these designs. We put a genetic programming system on top of that, and the design was fed back down to the professional analysis software. It concluded our AI design was 30% better than anything the contractor had ever built. It turned out that the genetic algorithm program had presented to the analysis program an airplane with negative wing load, which gave us negative drag, etc. So you have to be sure that your AI is constrained to the assumptions built into the analysis program.

Editor's note: Further discussion with the audience following Ben's introduction to AI included details of data analysis, how to get neural nets to learn more from fewer training examples and an acknowledgement that we do not know how many training examples are needed for different levels of complexity. There was also a brief list given of different software packages available for molecular design and analysis, from a priori quantum mechanics to molecular dynamics, with their respective strengths and weaknesses. It was also pointed out that is difficult to integrate genetic programming design with programs for simulations. An example of the success of such an effort is Rosetta, "The premier suite for macromolecular modeling". The developers of RosettaDesign won the Foresight Institute 2004 Feynman Prizes in Nanotechnology, Theory Category:

Dr. David Baker of the University of Washington, Department of Biochemistry, and Dr. Brian Kuhlman, of the University of North Carolina, Dept of Biochemistry and Biophysics, received the theory prize for their development of RosettaDesign, a program that has a high success rate in designing stable protein structures with a specified backbone folding structure. Their work includes the design of the first protein to be constructed with a backbone fold not observed in nature; in experimental testing, the novel backbone structure was found to be extremely stable and to match the predicted structure with atomic-level accuracy. Their work marks a milestone on a path to molecular machine systems. Professor Baker has made RosettaDesign freely available to the research community.

Artificial Intelligence for Nanoscale Design

A proposal sketch based on the Foresight Institute Research Workshop series

Allison Duettmann | James B. Lewis | Foresight Institute