Outline

• Computational Models
  • The ‘massive concurrency’ of molecular soups

• Discrete-state Molecular Systems
  • Combinatorial verification of (DNA) Chemical Reaction Networks

• Continuous-state Molecular Systems
  • Morphisms of Chemical Reaction Networks that preserve kinetics
Computational Models
A computational model

• Molecular ‘Soups’
  • Molecules randomly collide and can change state or composition.
    • Can we compute with that?
  • Based on the classical atomic theory of matter
    • probability of collision independent of location (“well-mixed” / “totally connected”)

• Related to:
  • For “small number of agents” (macroscopic systems):
    • Process Algebra, Petri Nets
  • For “large numbers of agents” (microscopic systems):
    • Population Protocols [Angluin et al.], Amorphous Computing [Abelson et al.]
    • Swarm Intelligence – Ant Colonies, Epidemiology, Morphology, Chemistry
A notion of algorithm

- Data as populations
  - Inputs and outputs are composed of uniform populations of agents that do not have an identity
  - Algorithms emerge from the ‘dumb’ interactions of ‘simple’ agents

- In computing
  - Mostly assuming discrete or nondeterministic time

- In science and nature
  - Mostly assuming stochastic or continuous time
  - Stochastic because interactions typically correspond to random collisions or chance meetings
A mathematical model

- Continuous-Time (Discrete-Space) Markov Chains
  - Also underlies chemistry via the Chemical Master Equation (changes of probabilities of discrete states over continuous time).
  - In the limit of infinite molecules at finite concentration, it converges to the deterministic continuous-state continuous-time (ODE) model.

- NOT a probabilistic (-only) model
  - Probabilities emerge from the stochastic structure (the underlying DMC), but are not primary. We are in continuous time and we care about how long things take.
  - Non-determinism exists only in the form of ‘quantitative races’: who is faster is more likely to win. There is no speed-independent probability.
  - Interleaving holds by the Markov axiom: no two events ever happen at the same time.

- What can we compute in this model?

(finite or infinite)
Programming Languages

- **Reaction-Based** \((A + B \rightarrow C + D)\) (Chemical Reactions)
  - Finite set of species (no polymerization): finite Markov chains.

- **Interaction-Based** \((A = !c. B)\) (Process Algebra)
  - Reduces combinatorial complexity of models by sharing *channels* between submodels.

- **Rule-Based** \((A{-}\colon B\{p\} \rightarrow A\{p\}\colon B{-}\)) (Logic, Graph Rewriting)
  - A *rule* is a reaction in a partially unspecified context.
  - Further reduces model complexity by abstracting over context.
  - Compatible with informal descriptions of biochemical events ("narratives").

- **Relationships**
  - The latter two can be translated to each other.
  - When they can be translated to the first, they may introduce an *extremely large* number of species.
Basic Results

- The class of functions ‘over individuals’ that are computable
  - A finite number of chemical reactions can encode Turing machines (only) up to an arbitrarily small uniform error bound. “Approximately Turing-Complete”. [1,2]
  - With polymerization, fully Turing completeness can be achieved.
  - But all these rely on ‘single-molecule populations’ that are difficult to achieve.

- The class of predicates ‘over populations’ that are ‘stably computable’ (population protocols)
  - Semi-linear predicates (first-order theory of (\(\mathbb{N}, +, <\)). [3]
  - If you cannot distinguish individual molecules, you are much more restricted.

Semantics of Chemistry (Chemical Kinetics)

- A connection with the theory of concurrency

More Languages & Models

- Gene Networks
  - Synchronous Boolean networks
    - Stewart Kauffman, etc.
  - Asynchronous Boolean networks
    - René Thomas, etc.

- Protein Networks
  - Process Algebra (stochastic $\pi$-calculus etc.)
    - Priami, Regev-Shapiro, etc.
  - Graph Rewriting (kappa, BioNetGen etc.)
    - Danos-Laneve, Fontana & al., etc.

- Membrane Networks
  - Membrane Computing
    - Gheorghe Păun, etc.
  - Brane Calculi
    - Luca Cardelli, etc.
Challenges in Discrete-State Molecular Systems

In collaboration with:
Microsoft Biological Computation Group
U.Oxford PRISM group
U.Washington Seelig Lab
‘Writing’ Molecular Programs

- Chemistry is not a computational science
  - We can read (nature’s) molecular programs, but we cannot write them (in general)!
  - We cannot find molecules that do whatever we want them to do!

- But we can fake it (encode it)
  - Find some ‘universal molecules’ that we can build, and that can do what all other molecules, real or hypothetical, can do.
  - Ok, not quite ‘do’, but ‘behave like’ any other molecules.

- With DNA
  - These are molecules we can read and write!
  - The folding problem for DNA/RNA is solvable, and they can be produced on industrial scale.

Soloveichik, D., Seelig, G., Winfree, E., 
Why write molecular programs?

- Non-goals
  - Not to solve NP-complete problems with large vats of chemicals (even massive concurrency does not help!)
  - Not to replace silicon-based technology
    DNA is slow(er), but compatible with life processes

- Bootstrapping a programmable carbon-based technology
  - To precisely control the organization and dynamics of matter and information at the molecular level
    - Nanotechnology
    - Medicine and Biology
  - DNA is “just” the most convenient material for the task
    - It is an information-bearing programmable material; other such materials are actively being developed
Domains

- Subsequences on a DNA strand are called **domains**
  - *provided* they are “independent” of each other

- Differently named domains must not **hybridize**
  - With each other, with each other’s complement, with subsequences of each other, with concatenations of other domains (or their complements), etc.
Short Domains

Reversible Hybridization

DNA double strand
Long Domains

Irreversible Hybridization
Strand Displacement

“Toehold Mediated”
Strand Displacement

Toehold Binding
Strand Displacement

Branch Migration
Strand Displacement

Displacement
Strand Displacement

Irreversible release
Bad Match

\[ t \quad x \quad z \]

\[ t \quad x \quad y \]
Bad Match
Bad Match
Bad Match

Cannot proceed
Hence will undo
Two-Domain Architecture

• Signals: 1 toehold + 1 recognition region

\[
\begin{array}{c}
\text{t} \\
\text{x}
\end{array}
\]

• Gates: “top-nicked double strands” with open toeholds

\[
\begin{array}{cccc}
\text{t} & \text{x} & \text{t} & \text{y} & \text{t}
\end{array}
\]

Garbage collection “built into” the gate operation

---

Two-Domain DNA Strand Displacement

Luca Cardelli

Plasmidic Gate Technology

- Synthetic DNA is length-limited
  - Finite error probability at each nucleotide addition, hence ~ 200nt max
- Bacteria can replicate plasmids for us
  - Loops of DNA 1000’s nt, with extremely high fidelity
  - Practically no structural limitations on fan-in/fan-out

Only possible with two-domain architecture
Transducer
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

Input

Join half

Fork half

**ta** is a *private* signal (a different ‘a’ for each xy pair)
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

Active waste

$x \quad t$

$t \quad a$

$t \quad x \quad t \quad a \quad t \quad a$

$y \quad t$

$x \quad t \quad y \quad t \quad a \quad t$
Transducer $x \rightarrow y$
So far, a **tx signal** has produced an **at cosignal**. But we want signals as output, not cosignals.
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$
Here is our output \textbf{ty signal}.
But we are not done yet:
1) We need to make the output irreversible.
2) We need to remove the garbage.
We can use (2) to achieve (1).
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$
Transducer $x \rightarrow y$

Done.

N.B. the gate is consumed: it is the energy source
(no proteins, no enzymes, no heat-cycling, etc.; just DNA in salty water)
Transducer → y
General $n \times m$ Join-Fork $= A_1 + \ldots + A_n \rightarrow B_1 + \ldots + B_m$ 

- Easily generalized to 2+ inputs (with 1+ collectors).
- Easily generalized to 2+ outputs.

Figure 9: 3-Join $J_{wxyz} | tw | tx | ty \rightarrow tz$: initial state plus inputs $tw, tx, ty$. 
With that, we can ‘implement chemistry’

• That is, we can implement *arbitrary* chemistry ...
  • ... by using *specific* (DNA) chemistry
  • ... up to an equivalence (same approximate kinetics, up to time dilation)

• Computing power equivalent to Stochastic Petri Nets
  • Not Turing complete, but as good as chemistry itself.
  • The correspondence is not completely trivial: gates are consumed by activation, hence a persistent Petri net transition requires a stable population of gates.
  • Many other mechanisms are expressible with Petri Nets like Boolean networks and state machines
Challenges of Correct Design: Proofs

- Does the two-domain architecture correctly implement Stochastic Petri Nets (and chemistry)?
  - A rather difficult problem (which I left open). By modelchecking we can verify specific constructions, but only for limited range of inputs.
  - This was only recently settled using techniques from the theory of concurrency (serializability):

  Matthew R. Lakin, Andrew Phillips, and Darko Stefanovic, Modular verification of DNA strand displacement networks via serializability analysis, in International Conference on DNA Computing and Molecular Programming, Springer Verlag, September 2013
Simulation

- Stochastic
- Deterministic
State Space Analysis

INITIAL STATE:

TERMINAL STATE:
Modelchecking

- PRISM probabilistic modelchecker
Verification

- Quantitative theories of system equivalence and approximation.
Scaling Up DNA Circuits

• Can verification catch up?

“In addition to biochemistry laboratory techniques, computer science techniques were essential.”

“Computer simulations of seesaw gate circuitry optimized the design and correlated experimental data.”
Challenges in Continuous-State Molecular Systems

In collaboration with:
Attila Csikász-Nagy
and thanks to David Soloveichik
Networks

- Informal ideas in Biology
  - Usually communicated by some kind of network or graph
  - These networks are often at best ambiguous [Kitano]

- Many kinds of networks, including:
  - Chemical Reaction Networks (species A becomes species B and C)
  - Influence Networks (species A promotes or inhibits species B)

- Networks convey meaning
  - Can network relationships convey meaning too?
Mutual Inhibition

- A recent paper suggests that all cellular switches in all phases of the cell cycle follow (abstractly) a mutual inhibition pattern:

- Also found in other areas (cell polarity establishment):
Cell Cycle Switch Network

- A recent paper presents a more complete view of the classical cell cycle switch
Network Emulation: NCC to MI

- For *any* initial state of MI we can find *some* initial state of NCC (actually by copying the state of MI) such that NCC *exactly* emulates MI.

Why does this work so well?

- Initialize z,r,p identically to z;
- Initialize y,q,s identically to y.
Network Emulation: MI to AM

- For chosen initial conditions of MI, the (6) trajectories of MI emulate those (3) of AM:

  \[ \sim y, z \rightarrow x \]

  (6 species on 3 trajectories)

  (3 species)

  initialize \( \sim y, z \), identically to \( x \)
Influence Network Notation

- Catalytic reaction
  \[ x \rightarrow y = x \rightarrow y \]
  \[ z \text{ is the catalyst} \]
  \[ x + z \rightarrow z + y \]

- 'Double kinase-phosphatase' motif

influence node  catalytic node
MI to AM mapping in detail

Homomorphic mapping

Initial conditions:
- $z_0 = y_2 = x_0$
- $z_1 = y_1 = x_1$
- $z_2 = y_0 = x_2$

Any initial conditions
Network Emulation Composes: NCC to AM

- For chosen initial conditions of NCC, the (18) trajectories of NCC emulate those (3) of AM.

This works also for GW, but not for the original CC.

The new cell cycle switch can emulate AM exactly. For any initial conditions of AM. And for any rates of AM. Why?
An Analytical Theory of Network Emulation

- An emulation is an “implementation”
  - “for every input produces the same output” ➔
    “for every initial conditions produces the same trajectories”
  - A refined network that works just as well as the coarser network
    *in the context* of the inputs of the coarser network (not arbitrary inputs)

- When can a network emulate another one?
  - Theories of behavioral equivalence and behavioral approximation,
    e.g. like in process algebra, are still lacking in this quantitative field
  - So we look at the continuous-state semantics of these networks,
    and see what we can do there
Chemical Reaction Networks

- A CRN is a pair \((S, R)\) where
  - \(S = \{s_1, \ldots, s_n\}\) is a finite set of species
  - \(R = \{r_1, \ldots, r_m\}\) is a finite set of reactions over \(S\)

- Reactions \(r = (\rho, \pi, k)\) written \(\Sigma_{s \in S} \rho_s \cdot s \rightarrow^k \Sigma_{s \in S} \pi_s \cdot s\)
  - Ex.: \(r = 2A + B \rightarrow^k A + 3C\)
  - with \(\rho_A = 2, \rho_B = 1, \rho_C = 0\)
    \(\pi_A = 1, \pi_B = 0, \pi_C = 3\)

- The stoichiometry of a species \(s\) in a reaction \(r\) is:
  \[
  \eta(s, (\rho, \pi, k)) = \pi_s - \rho_s \quad \text{net stoichiometry} \quad \eta(A, r) = -1 \\
  \varphi(s, (\rho, \pi, k)) = k \cdot (\pi_s - \rho_s) \quad \text{(instantaneous) stoichiometry} \quad \varphi(A, r) = -k
  \]
Species Maps and Reaction Maps

- A *species map* is a map \( m \in S \rightarrow \hat{s} \)
- Ex: \( m(s_0) = m(s_1) = \hat{s} \)
- It induces a canonical *reaction map* \( R \rightarrow \hat{R} \)
- Ex: \( m(s_0 + s_1 \rightarrow^1 s_1) = 2\hat{s} \rightarrow^1 \hat{s} \)
- Where \( m(\rho, \pi, k) = (m(\rho), m(\pi), k) \)
- And \( m(\rho) \) (similarly \( m(\pi) \)) is the sum over fibers:
  \[
m(\rho)_{\hat{s}} = \Sigma_{s \in m^{-1}(\hat{s})} \rho_s
  \]
  in case two species in the same reaction are mapped to the same species.
CRN Morphisms

- A **CRN morphism** is a map $m \in (S, R) \rightarrow (\hat{S}, \hat{R}) = (m_S, m_R)$ with $m_S \in S \rightarrow \hat{S}$ and $m_R \in R \rightarrow \hat{R}$.

- We are interested in morphisms that are *not* injective, that represent *implementations*.
CRN Homomorphisms

- \( m \in (S, R) \rightarrow (\hat{S}, \hat{R}) \) is a **CRN homomorphism** if \( m_R \) is determined by \( m_S \):
  \[
m_R(\rho, \pi, k) = (m_S(\rho), m_S(\pi), k)
  \]

- Ex:
  - \( r_0: \ m_R(s_0, s_1, k) = (\hat{s}_0, \hat{s}_1, k) = (m_S(s_0), m_S(s_1), k) \)
  - \( r_1: \ m_R(s_0, s_2, k) = (\hat{s}_0, \hat{s}_1, k) = (m_S(s_0), m_S(s_2), k) \)

- It implies that **for each reaction it preserves stoichiometry summed over species fibers**
  \[
  \forall \hat{s} \in \hat{S}. \ \forall r \in R. \ \Sigma_{s \in m^{-1}(\hat{s})} \varphi(s, r) = \varphi(\hat{s}, m(r))
  \]

- But \( \varphi(s_0, r_0) + \varphi(s_0, r_1) = -2k \neq -1k = \varphi(\hat{s}_0, \hat{r}_0) \) (see next slide)
CRN Stoichiomorphisms

- \( m \in (S, R) \rightarrow (\hat{S}, \hat{R}) \) is a **CRN stoichiomorphism** if for each species it preserves stoichiometry summed over reaction fibers

\[ \forall s \in S. \ \forall \hat{r} \in \hat{R}. \ \Sigma_{r \in m^{-1}(\hat{r})} \varphi(s, r) = \varphi(m(s), \hat{r}) \]

- This condition can be checked over the syntax of CRNs, without any consideration of their kinetics
  - Ex:
    - \( s_0, \hat{r}_0: \ \varphi(s_0, r_0) + \varphi(s_0, r_1) = 0 = \varphi(\hat{s}_0, \hat{r}_0) \)
    - \( s_1, \hat{r}_0: \ \varphi(s_1, r_0) + \varphi(s_1, r_1) = 1k = \varphi(\hat{s}_1, \hat{r}_0) \)
    - \( s_2, \hat{r}_0: \ \varphi(s_2, r_0) + \varphi(s_2, r_1) = 1k = \varphi(\hat{s}_2, \hat{r}_0) \)

- We will show that existence of a stoichiomorphism implies identical network kinetics (in certain conditions).
CRN Morphism Conditions

- Homomorphism consequence:
  \[ \forall \hat{s} \in \hat{S}. \ \forall r \in R. \ \Sigma_{s \in m^{-1}(\hat{s})} \varphi(s, r) = \varphi(\hat{s}, m(r)) \]

- Stoichiomorphism condition:
  \[ \forall s \in S. \ \forall \hat{r} \in \hat{R}. \ \Sigma_{r \in m^{-1}(\hat{r})} \varphi(s, r) = \varphi(m(s), \hat{r}) \]

- If \( m \) is an isomorphism (injective and surjective, with singleton fibers) then they both reduce to the simple property:
  \[ \forall s \in S. \ \forall r \in R. \ \varphi(s, r) = \varphi(m(s), m(r)) \]

- The above are generalization for when \( m \) is not injective.
Checking the Stoichiomorphism Condition

\[ m \in \text{MI} \rightarrow \text{AM} \]

\[ \forall s \in S. \ \forall \hat{r} \in \hat{R}. \ \Sigma_{r \in m^{-1}(\hat{r})} \varphi(s, r) = \varphi(m(s), \hat{r}) \]

\[ \varphi(y_0, m_{i_0}) + \varphi(y_0, m_{i_4}) = -1 = \varphi(x_2, a_{i_0}) \]

All unit rates (for simplicity)

This is both a homomorphism and a stoichiomorphism
CRN Kinetics

- A state of a CRN \((S, R)\) is a vector of concentrations for each species: \(\nu \in \mathbb{R}^+\).

- The mass action \([r] \in \mathbb{R}^+_S \rightarrow \mathbb{R}^+\) of a reaction \(r \in R\) is:

  \[
  [r]_\nu = [(\rho, \pi, k)]_\nu = \prod_{s \in S} \nu_s^{\rho_s} = \nu^\rho
  \]

- The differential system of a CRN \((S, R)\) is the map \(F \in \mathbb{R}^+ \rightarrow \mathbb{R}^S\) (for each state, gives the differential of concentration for each species):

  \[
  F(\nu)(s) = \Sigma_{r \in R} \varphi(s, r) \cdot [r]_\nu
  \]

- Normally written as a system of concentration ODEs, integrated over time:

  \[
  \frac{d\nu_s}{dt} = F(\nu)(s) = \Sigma_{(\rho, \pi, k) \in R} k \cdot (\pi_s - \rho_s) \cdot \nu^\rho
  \]
Kinetic Emulation

- A map $m \in (S, R) \to (\mathring{S}, \mathring{R})$ is a CRN emulation if the following holds for the respective differential systems $F, \hat{F}$:

  $$\forall \hat{\nu} \in \mathbb{R}^+ \mathring{S}. \forall s \in S. F(\hat{\nu} \circ m)(s) = \hat{F}(\hat{\nu})(m(s))$$

  (the derivative of $s$ in state $\hat{\nu} \circ m$ is equal to the derivative of $m(s)$ in state $\hat{\nu}$)

- It follows that for any initial state $\hat{\nu}$ of $(\mathring{S}, \mathring{R})$ there is an initial state $\nu (=\hat{\nu} \circ m)$ of $(S, R)$ such that the trajectory of any $s$ in $(S, R)$ is identical to (emulates) the trajectory of $m(s)$ in $(\mathring{S}, \mathring{R})$.

  (With minor caveats if $m$ is not surjective.)
Emulation Theorem

- Theorem: If $m$ is a CRN homomorphism and stoichiomorphism then it is a CRN emulation.

That is, for any initial conditions we can match trajectories.

- Actually, $m$ need not be a homomorphism for this to hold: it is enough for $m$ to be a reactant morphism and a stoichiomorphism. A reactant morphism agrees with the species map on the reactant species, but allows rates and product species to disagree. This allows a wider range of network mappings that preserve kinetics.
Change of Rates Theorem

- A change of rates for \((S, R)\) is bijection \(\iota \in (S, R) \rightarrow (S, R')\) such that \(\iota(S)\) is the identity and \(\iota(\rho, \pi, k) = (\rho, \pi, k')\).

- Theorem: If \(m \in (S, R) \rightarrow (\hat{S}, \hat{R})\) is a stoichiomorphism, then for any change of rates \(\hat{\iota}\) of \((\hat{S}, \hat{R})\) there is a change of rates \(\iota\) of \((S, R)\) such that \(\hat{\iota} \circ m \circ \iota^{-1}\) is a stoichiomorphism.

  - In fact, \(\iota\) changes rates by the ratio with which \(\hat{\iota}\) changes rates:
    \[\iota(\rho, \pi, k) = (\tilde{\rho}, \tilde{\pi}, \tilde{k})\] where \(m(\rho, \pi, k) = (\hat{\rho}, \hat{\pi}, \hat{k})\) and \(\hat{\iota}(\hat{\rho}, \hat{\pi}, \hat{k}) = (\tilde{\rho}, \tilde{\pi}, \tilde{k}')\).

- Corollary: If \(m \in (S, R) \rightarrow (\hat{S}, \hat{R})\) is a stoichiomorphism and homomorphism, then for any change of rates \(\hat{\iota}\) of \((\hat{S}, \hat{R})\) there is a change of rates \(\iota\) of \((S, R)\) such that \(\hat{\iota} \circ m \circ \iota^{-1}\) is an emulation.
Any Rates, Any Initial Conditions

- A stoichiomorphism $m \in (S, R) \to (\hat{S}, \hat{R})$ that is also a homomorphism, determines an emulation for any choice of rates of $(\hat{S}, \hat{R})$.

- Those emulations can match any initial conditions of any choice of rates of $(\hat{S}, \hat{R})$ with some initial conditions of some choice of rates of $(S, R)$.
Stoichiomorphism Zoo

(homomorphism and stoichiomorphism (transitive))
Interpretation of Stoichiomorphism

- Ignorance about initial conditions
  - We may not know the concentrations of species in the more complex network, but at least we know that if they satisfy certain conditions, then it behaves like the simpler network.

- Neutral paths in network space (evolution)
  - If an evolutionary event happens to be a stoichiomorphism, or close to it, it will not be immediately selected against, because it is “kinetically neutral”.
  - This allows the network to increase its complexity without kinetic penalty.
  - Later, the extra degrees of freedom can lead to kinetic differentiation.
  - But meanwhile, the organism can explore variations of network structure.

- Relationship to abstraction / coarse-graining
  - Stoichiomorphism are not about abstractions that preserve behavior, on the contrary, they are about concretions that preserve behavior.
  - They describe implementations of abstract specs, where the specs themselves may not be (biologically) implementable because of excessive demands on individual species.
Conclusions
Conclusions

- The promise of nanotechnology
  - Controlling matter and information in detail at the molecular scale
    - This can only be achieved by digital (combinatorial) techniques
  - Interfacing to natural (biological) systems, which often have analog properties
    - This usually involves using continuous modeling/techniques

- Discrete systems are hard to engineer
  - We need combinatorial analysis techniques that scale up (massively!)
  - We need verification and approximation techniques for massive concurrency

- Continuous systems are hard to understand
  - Calculus is the weapon of choice, but even there qualitative understanding is king
  - We need quantitative methods that support qualitative reasoning