Tutorial

The Computational Power of Chemical Reaction Networks

David Soloveichik

UCSF center for systems & synthetic biology

an NIGMS national systems biology center

Why Study CRNs?

- Fundamental model of chemical kinetics used in the natural sciences

$$CO_{2(aqueous)} + H_2O \rightleftharpoons H_2CO_3$$

- Fundamental model of population dynamics in ecology

- Sensor networks
  (population protocols)

- Fundamental mathematical structure:
  (vector addition systems, Petri nets
  commutative semigroups, bounded context-free
  languages, uniform recurrence equations)

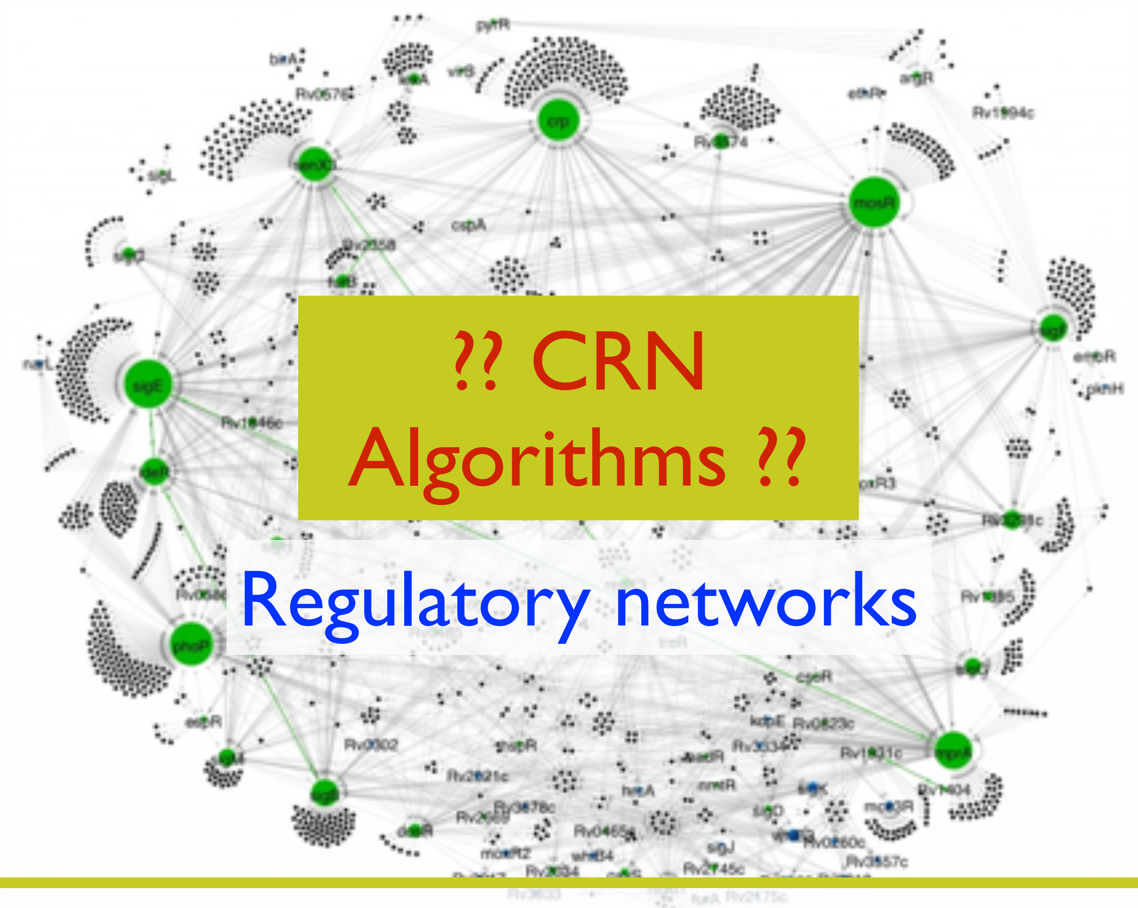
$$\Omega$$

Why Understand Computation with CRNs

- Embed programming into environments not compatible with traditional von Neumann computer architectures

Why Understand Computation with CRNs

- How do cells process information?



Question: How to make the computational perspective herein more relevant in biology?

Outline

Chemical Reaction Networks (CRNs)
Discrete Model

- *d* species and *r* reactions
- A state $x \in \mathbb{N}^d$ specifies the molecular counts of each species
- Reactions $\{rxn_1, \ldots, rxn_r\}$ where each reaction
  $rxn_i = (r_i, p_i, k_i) \in \mathbb{N}^d \times \mathbb{N}^d \times \mathbb{R}^+$
- Reaction rxn_i can occur in state x if $x - r_i \geq 0$. If reaction rxn_i occurs in state x, the state changes from x to $x - r_i + p_i$

Chemical notation:

$((1,1,0),\ (1,0,1),\ k_1) \quad \longleftrightarrow \quad A + B \xrightarrow{k_1} A + C$

$((0,0,1),\ (2,0,0),\ k_2) \quad \longleftrightarrow \quad C \xrightarrow{k_2} A + A$

Chemical Reaction Networks (CRNs)
Discrete Model

- Start in some initial state *x* in solution volume v
- The system evolves via a continuous time Poisson process:

Suppose current state $x=(a, b, c, \ldots)$.

The time until next reaction is exponentially distributed with rate $\sum prop_i$

The probability that the next reaction is rxn_j is $prop_j / \sum prop_i$

reaction type	$prop_i$
$A \xrightarrow{k} \ldots$	$k \cdot a$
$A + B \xrightarrow{k} \ldots$	$k \cdot a \cdot b \ / \ v$
$A + A \xrightarrow{k} \ldots$	$k \cdot a \cdot (a-1) \ / \ (2v)$

McQuarrie 1967, van Kampen, Gillespie 1977, etc

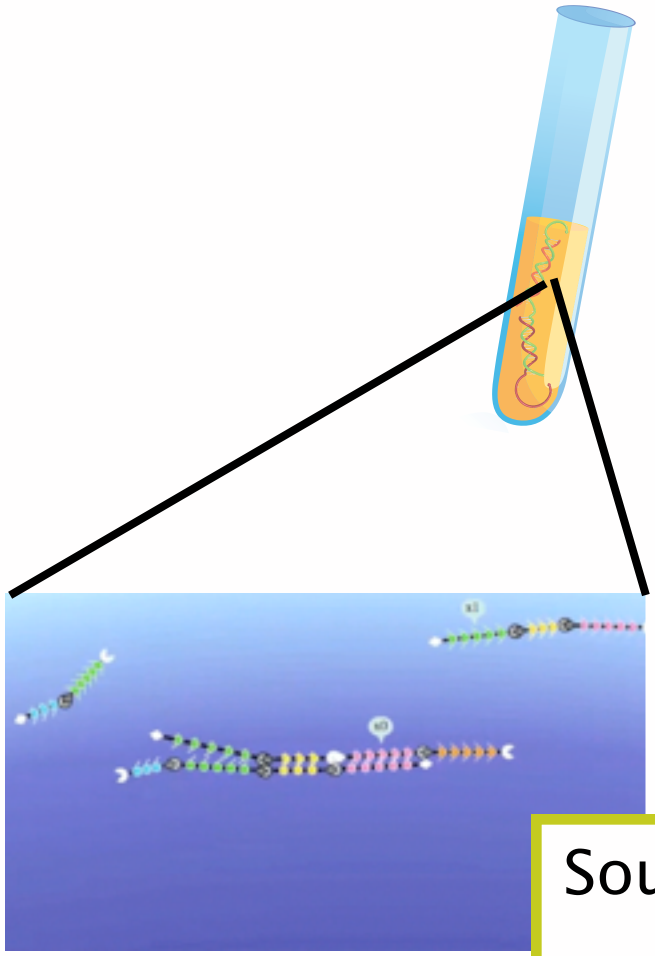
Scaling up from the stochastic to the deterministic model

Increase solution volume *v* and the molecular counts of all species such that for each species $\#X/v$ stays constant.

In the limit $v \to \infty$, we get the deterministic system described by ODEs.

Does every CRN have a molecular realization?

Molecular Realization of CRNs with Strand Displacement Cascades



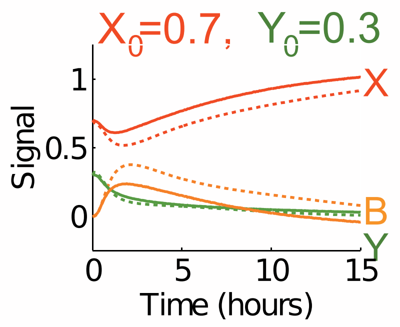
- Soloveichik, Seelig, Winfree "DNA as a Universal Substrate for Chemical Kinetics" *PNAS* (2010)
- Qian, Soloveichik, Winfree, "Efficient Turing-Universal Computation With DNA Polymers" *DNA* 16 (2010)
- Chiniforooshan, Doty, Kari, Seki "Scalable, Time-responsive, Digital, Energy-Efficient Molecular Circuits Using DNA Strand Displacement" *DNA 16* (2010)
- Cardelli, "Strand Algebras for DNA Computing" *Nat Comp* (2011)
- Cardelli, "Two-Domain DNA Strand Displacement" *Math Structs CS* (2013)

Source of energy and mass: auxiliary species

Does every CRN have a molecular realization?

Molecular Realization of CRNs with Strand Displacement Cascades

$$X + Y \xrightarrow{k1} B + Y$$
$$X + Y \xrightarrow{k2} X + B$$
$$B + X \xrightarrow{k3} X + X$$
$$B + Y \xrightarrow{k4} Y + Y$$



Chen, Dalchau, Srinivas, Phillips, Cardelli, Soloveichik, Seelig, Winfree "Programmable Chemical Controllers Made From DNA" *Nature Nanotechnology* (2010)

Outline

Model

Semilinear computation

Turing-universal computation

Computation speed

Some open questions

Dichotomies of Computation in CRNs

discrete vs continuous: Which model of CRNs?

uniform vs non-uniform: Is a single CRN supposed to handle all inputs, or do we add reactions for larger inputs?

deterministic vs probabilistic: Is the correct output guaranteed or merely likely?

halting vs stabilizing: Does the CRN "know" when it has finished?

Dichotomies of Computation in CRNs

discrete vs continuous: Which model of CRNs?

uniform vs non-uniform: Is a single CRN supposed to handle all inputs, or do we add reactions for larger inputs?

halting: irreversibly produce N or Y

stabilizing: eventually stabilize to N or Y

istic: Is the correct

kely?

halting vs stabilizing: Does the CRN "know" when it has finished?

Dichotomies of Uniform Computation

	halting	**stabilizing**
deterministic	(finite)	semilinear
probabilistic	Turing-universal	Δ_2^0

Predicate Computation (Example)
Deterministic, Stabilizing

Start with $1Y$ and input amounts of $X1, X2$
Eventually stabilizes to a state with Y (YES) or N (NO)

Predicate: $X2 \geq X1$?

$$X2 + N \rightarrow Y$$
$$X1 + Y \rightarrow N$$

Predicate: X1 == X2?

$$X1 + X2 \rightarrow Y$$
$$Y + N \rightarrow Y$$
$$X1 + Y \rightarrow X1 + N$$
$$X2 + Y \rightarrow X2 + N$$

Predicate Computation
Deterministic, Stabilizing

initial state: input counts of input species, fixed amounts of other species

output of a state:

contains Y but not N	$\leftrightarrow$	YES
contains N but not Y	$\leftrightarrow$	NO
otherwise	$\leftrightarrow$	undefined

output-stable states: states with YES or NO output such that all states reachable from them have the same output

deterministic, stabilizing computation:
For any input, a correct output-stable state is reachable from any reachable state. (Implies that incorrect output-stable states are not reachable.)

An Impossibility Result
Deterministic, Stabilizing

Claim: Predicate $X2 \geq (X1)^2$ cannot be computed in this way

A proof sketch:

- A few facts about recognizing stability
- How states can be truncated to preserve stability
- A Pumping Lemma

Angluin, Aspnes, Eisenstat, "Stably computable predicates are semilinear" (2006)

Recognizing Unstable States

Fact 0: Let x,y,z be states. If $x \Longrightarrow y$ then $x+z \Longrightarrow y+z$

Define: State x is unstable if $x \Longrightarrow y$ such that y has opposite output of x, or y has undefined output (and x, y have at least one molecule of output species)

Fact 1: If x is unstable, then any $y \geq x$ is also unstable.

Fact 2: Any set in $\mathbb{N}^d$ has a finite number of minimal elements (Dickson's Lemma).

Claim 1: For any CRN, there is a finite set of states $U=\{u_1,\ldots,u_m\}$ such that:
x is unstable iff $x \geq u_i$ for some $u_i \in U$

Truncations that Preserve Stability

Assume the CRN computes correctly (deterministic, stabilizing).

Choose a "threshold" τ larger than the amount of any species in any state in U.

Claim 2: Suppose states $x \leq y$, x is YES-output-stable, and y is larger than x only on species whose count is at least τ in x. Then y can't be reached from NO-input.

Pf: By Claim 1, y can't be unstable. Thus y can't reach a NO voter. y contains Y species so by CRN correctness can't be reachable from NO input.

Pumping Lemma 1/3

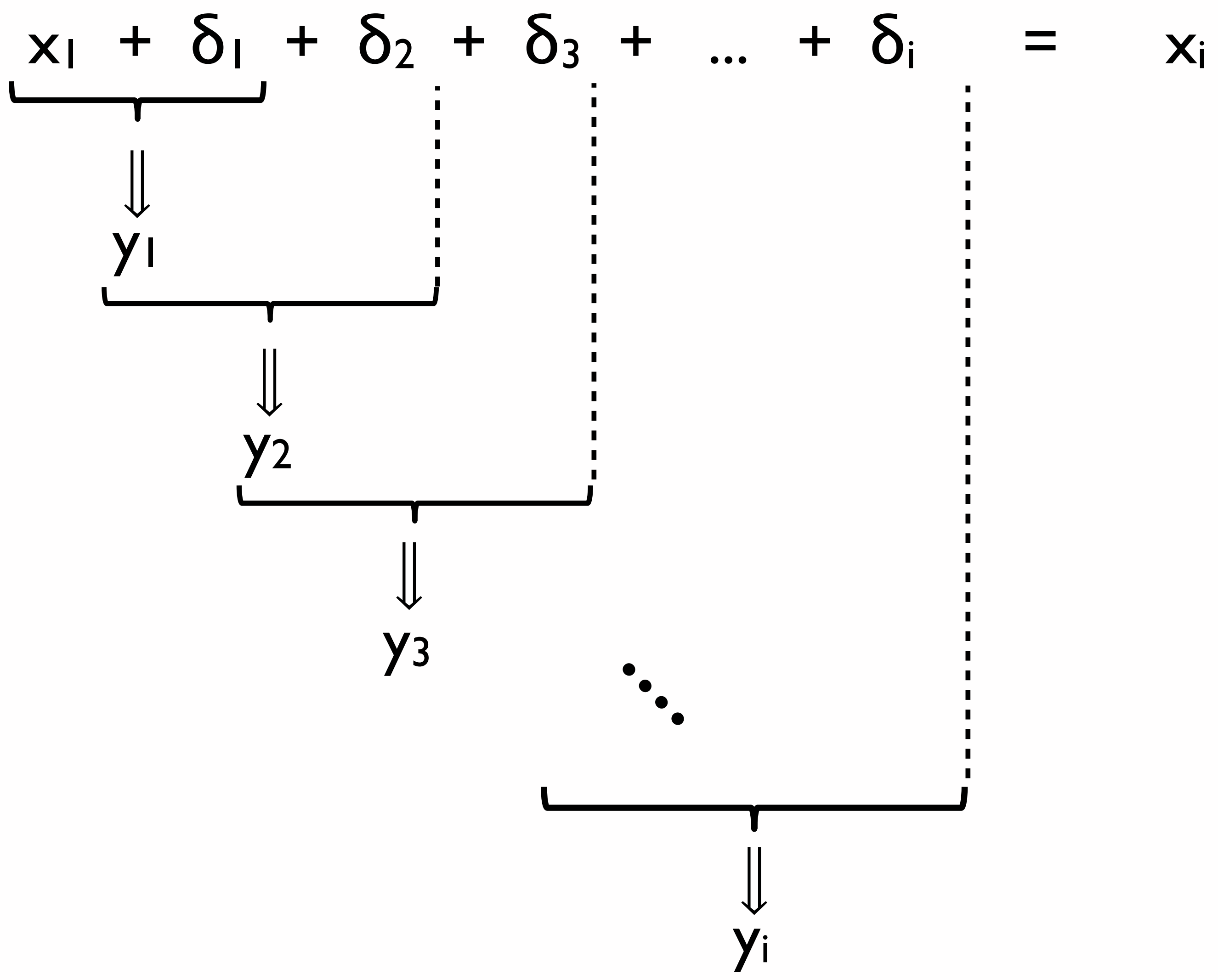
Consider any infinite increasing sequence of YES inputs and the corresponding (non-decreasing) sequence of input states $x_1, x_2, \ldots$

Let $\delta_i = x_{i+1} - x_i$

We know that every $x_i \Longrightarrow$ [some YES-output-stable state].

But we take a more specific path.

Pumping Lemma 2/3


$x_1 + \delta_1 + \delta_2 + \delta_3 + \ldots + \delta_i = x_i$
y_1
y_2
y_3
y_i

Claim 3: all y_i can be YES-output-stable.

W.l.o.g. y_i can be non-decreasing (Dickson's Lemma)

Pumping Lemma

There are $y_i \leq y_k$ that have the same counts of all species $< \tau$.

Recall: $y_i + \delta_{i+1} + \ldots + \delta_k \Longrightarrow y_k$

This path converts input species $(\delta_{i+1} + \ldots + \delta_k)$ into species that are at least τ in y_k.

Add $(\delta_{i+1} + \ldots + \delta_k)$ to y_k and take the same path. We get a new state z that cannot be reached from a NO-output (by Claim 2). z can be reached from $x_k + (\delta_{i+1} + \ldots + \delta_k)$, so that must be a YES-input.

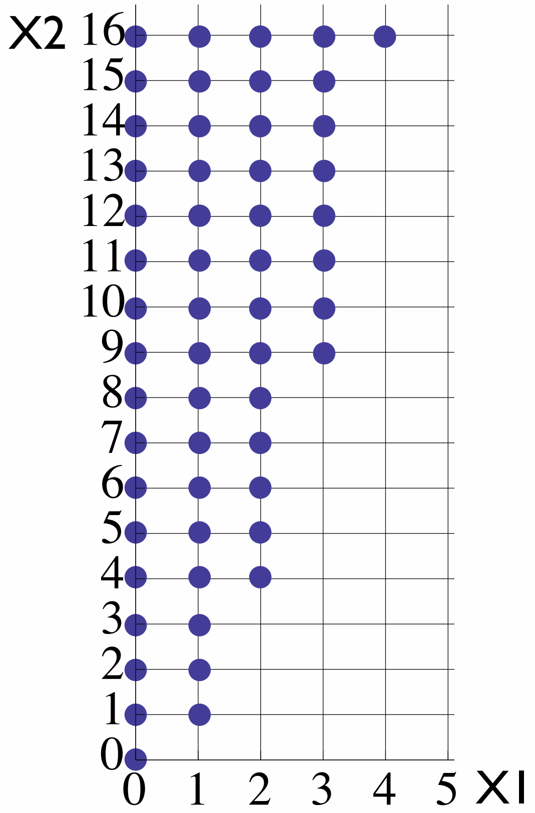
Repeat.

An Impossibility Result

Deterministic, Stabilizing

Lemma: If there is an infinite sequence of (distinct) YES inputs $x_1, x_2, \ldots$, then there are $x_i < x_k$ and such that all of $\{x_i + n \cdot (x_k - x_i) \mid n \in \mathbb{N}\}$ is YES also.

Ex: Predicate $X2 \geq (X1)^2$

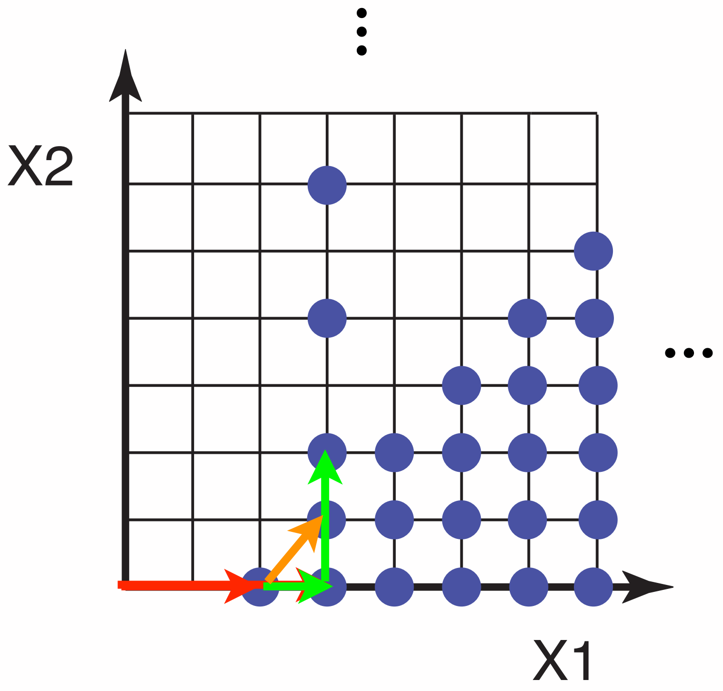


Definition of Semilinear Sets

A set $A \subseteq \mathbb{N}^d$ is **linear** if there exist vectors $\mathbf{b}, \mathbf{u_1}, \ldots, \mathbf{u_p} \in \mathbb{N}^d$ such that

$$A = \{\mathbf{b} + n_1\mathbf{u_1} + \cdots + n_p\mathbf{u_p} \mid n_1, \ldots, n_p \in \mathbb{N}\}$$

A is **semilinear** if it a finite union of linear sets.



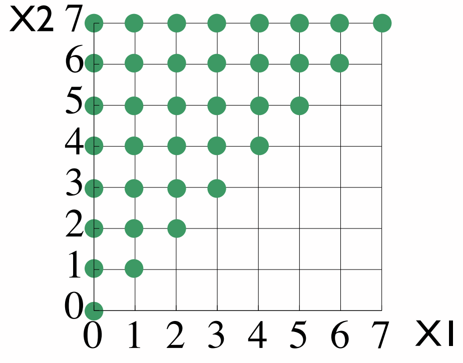
{(3,0)+n₁·(0,2) | n₁∈N} ∪ {(2,0)+n₁·(1,0)+n₂·(1,1) | n₁,n₂∈N}

Computational Power of Deterministic, Stabilizing CRNs

Theorem: Predicates computable in this manner are exactly the semilinear predicates

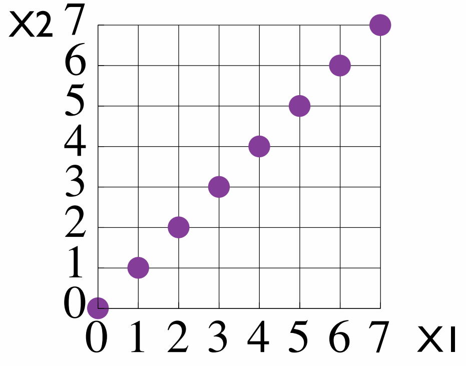
Angluin, Aspnes, Eisenstat, "Stably computable predicates are semilinear" (2006)

Predicate: $X2 \geq X1$?



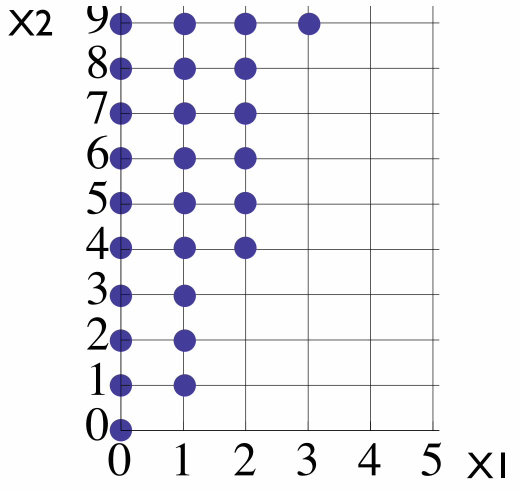
$\{n_1 \cdot (0,1) + n_2 \cdot (1,1) \mid n_1, n_2 \in \mathbb{N}\}$

Predicate: $X1 == X2$?



$\{n_1 \cdot (1,1) \mid n_1 \in \mathbb{N}\}$

Predicate: $X1 \geq (X2)^2$?



not semilinear

Function Computation (Example)
Deterministic, Stabilizing

$f(x)=2x$

$X \rightarrow Y+Y$

$f(x_1, x_2)=\min(x_1, x_2)$

$X1+X2 \rightarrow Y$

$f(x_1, x_2)=x_1+x_2$

$X1 \rightarrow Y$

$X2 \rightarrow Y$

$f(x_1, x_2)=\max(x_1, x_2)$

$X1 \rightarrow L1+Y$

$X2 \rightarrow L2+Y$

$L1+L2 \rightarrow K$

$Y+K \rightarrow \varnothing$

$f(x_1, x_2)=x_1$ if $x_1>x_2$ and 0 otherwise

$X1 \rightarrow N+L$

$L+X2 \rightarrow \varnothing$

$L+N \rightarrow L+Y$

$X2+Y \rightarrow X2+N$

$N+Y \rightarrow N+N$

Computational Power of Deterministic, Stabilizing CRNs

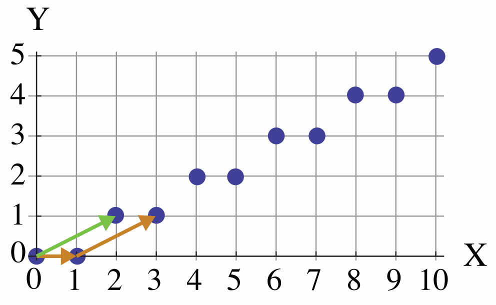
Theorem: Functions computable in this manner are exactly those with semilinear graphs

Chen, Doty, Soloveichik, "Deterministic Function Computation with Chemical Reaction Networks" (2013)

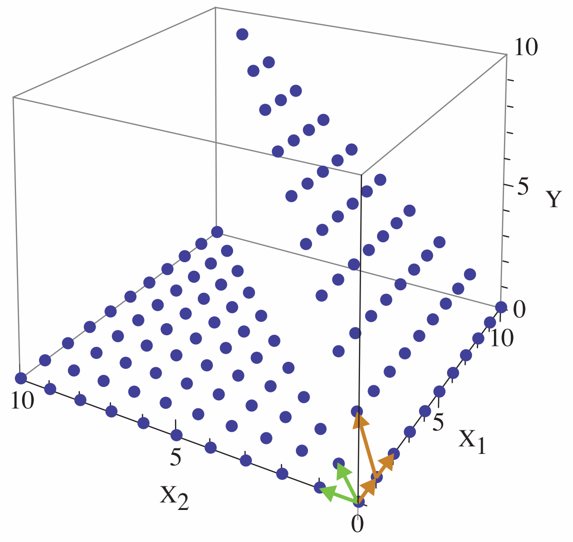
$f(x) = \lfloor x/2 \rfloor$

$f(x_1, x_2) = x_2$ if $x_1 > x_2$ and 0 otherwise

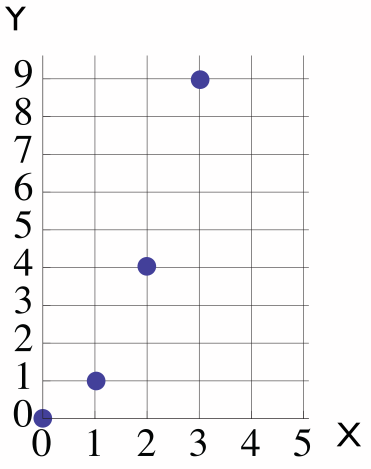
$f(x) = X^2$



$\{n_1 \cdot (2,1) \mid n_1 \in \mathbb{N}\} \cup$
$\{(1,0) + n_1 \cdot (2,1) \mid n_1 \in \mathbb{N}\}$



$\{n_1 \cdot (1,1,0) + n_2 \cdot (0,1,0) \mid n_1, n_2 \in \mathbb{N}\} \cup$
$\{(1,0,0) + n_1 \cdot (1,1,1) + n_2 \cdot (1,0,0) \mid n_1, n_2 \in \mathbb{N}\}$



not semilinear

Outline

Model

Semilinear computation

Turing-universal computation

Computation speed

Some open questions

Dichotomies of Uniform Computation

	halting	**stabilizing**
deterministic	(finite)	semilinear
probabilistic	Turing-universal	Δ_2^0

Computational Power of Probabilistic, Halting CRNs

Show Turing Universality by simulating **Register Machines** (aka Minsky Counter Machines)

- fixed number of registers, each holding nonnegative integer
- two kinds of instructions:

i: inc(r,j) $\longleftrightarrow$	increment register r and go to instruction j
i: dec(r,j,k) $\longleftrightarrow$	if register r is >0, then decrement it and go to instruction j; otherwise, go to instruction k

Example: f(x)=x^2

- 4 registers, 9 instructions
- start with input in reg R1
- halt with output in reg R4

```
1: dec(R1,2,9)      6: inc(R4,4)
2: inc(R2,3)        7: dec(R1,8,9)
3: inc(R3,4)        8: inc(R2,7)
4: dec(R2,5,7)      9: dec(R3,4,halt)
5: inc(R1,6)
```

Computational Power of Probabilistic, Halting CRNs

Register machines have a very natural CRN implementation:

One molecule of $S_1, \ldots, S_m$ to store the current instruction
The number of molecules of species R_r stores the value of register r

i: inc(r,j) $\longleftrightarrow$ $S_i \rightarrow R_r + S_j$

i: dec(r,j,k) $\longleftrightarrow$ $S_i + R_r \rightarrow S_j$
$S_i \rightarrow S_k$

Problem: second reaction may occur even if register is non-zero

Computational Power of Probabilistic, Halting CRNs

Register machines have a very natural CRN implementation:

One molecule of $S_1, \ldots, S_m$ to store the current instruction
The number of molecules of species R_r stores the value of register r

i: inc(r,j) $\longleftrightarrow$ $S_i \rightarrow R_r + S_j$

i: dec(r,j,k) $\longleftrightarrow$ $S_i + R_r \rightarrow S_j + A$
$C_1 + S_i \rightarrow S_k + C_3$

"clock" module:

$C_1 + A \rightarrow C_2 + A$
$C_2 \rightarrow C_1$
$C_2 + A \rightarrow C_3 + A$
$C_3 \rightarrow C_2$

Computational Power of Probabilistic, Halting CRNs

Theorem: Turing-universal computation is possible in this setting with the error probability bounded independently of the number of steps.

Soloveichik, Cook, Winfree, Bruck, "Computation with Finite Stochastic Chemical Reaction Networks" (2008)

Outline

Model

Semilinear computation

Turing-universal computation

Computation speed

Some open questions

Measuring Computation Speed “Fairly”

1. Fix largest rate constant (say k=1)

2. Volume v = O(maximum molecular count)

   This translates to upper-bounding concentrations in the deterministic limit.

   *For Turing-universal computation assume volume dynamically scales with total molecular count.

Speed of Turing-Universal Computation

Suppose we simulate a TM that runs in t steps and uses s bits of memory.

Two apparent sources of slowdown:

1. **Too many steps:** RM requires $\Omega(t\ 2^s)$ steps.

2. **Slow reactions:** Volume $v = \Omega$([count of A] + [count of register species]) $= \Omega(2^t + 2^s)$.
   Thus the worst case expected time for reaction $S_i + R_r \xrightarrow{1} S_j$ is $v = \Omega(2^t + 2^s)$.

Speed of Turing-Universal Computation

Surprisingly, both issues can be overcome for polynomial and even linear time simulation.

Angluin, Aspnes, Eisenstat, "Fast computation by population protocols with a leader" (2006)

Soloveichik, Cook, Winfree, Bruck, "Computation with Finite Stochastic Chemical Reaction Networks" (2008)

Implies that likely there is no general way to speed up "tau-leaping".

Soloveichik, "Robust Stochastic Chemical Reaction Networks and Bounded Tau-Leaping" (2009)

Speed of Semilinear Computation (Examples)

n=number of input molecules
volume $v = \Theta(n)$

Compute $f(x) = 2 \cdot x$

input: X
output: Y

$$X \xrightarrow{1} Y + Y$$

expected time to finish:

$$\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1} = \Theta(\log n)$$

fast!

Speed of Semilinear Computation (Examples)

n=number of input molecules
volume *v* = $\Theta(n)$

Compute $f(x) = \lfloor x/2 \rfloor$

input: *X*
output: *Y*

$$X + X \xrightarrow{1} Y$$

expected time to finish:

$$\frac{2n}{n(n-1)} + \frac{2n}{(n-1)(n-2)} + \cdots + \frac{2n}{2 \cdot 1} = \Theta(n)$$

slow!

Speed of Semilinear Computation (Examples)

n=number of input molecules
volume $v = \Theta(n)$

Compute $f(x_1,x_2) = \begin{cases} x_2 \text{ if } x_1>0 \\ 0 \text{ otherwise} \end{cases}$

input: X_1, X_2
output: Y

expected time to finish: $O(\log n)$

$$X_1 + X_2 \xrightarrow{1} X_1 + Y$$
$$Y + X_2 \xrightarrow{1} Y + Y$$

hint: consider two cases:
$x_1 < x_2$, and $x_2 \geq x_1$

fast!

Speed of Semilinear Computation

n=number of input molecules
volume $v = \Theta(n)$

Theorem: Every semilinear predicate/function can be deterministically computed by a chemical reaction network that stabilizes in expected time O(polylog(n)) (ie polynomial in the number of bits to write the input in binary)

Trick:

- Combine fast probabilistic, halting computation with slow deterministic, stabilizing computation. If the fast computation is correct then the correct output stabilizes quickly. Otherwise, slow computation corrects error.
- Error probability of fast computation small enough that the overall expected time is almost that of the fast computation

Angluin, Aspnes, Eisenstat, "Fast computation by population protocols with a leader" (2006)
Chen, Doty, Soloveichik, "Deterministic Function Computation with Chemical Reaction Networks" (2013)

Outline

Model

Semilinear computation

Turing-universal computation

Computation speed

Some open questions

Some Open Problems for Theory

- Computational complexity of reachability
- Better characterize non-uniform computation
- Fuel and energy use
  fundamental discoveries about the thermodynamics of computation?
- Models of cellular regulatory networks
  relevant insights for systems/synthetic biology?

Computational complexity of reachability

Exact state reachability

Problem: Given a CRN, states x and y, figure out if y is reachable from x.
At least exponential space: Lipton 1976
Decidable: Mayr 1981, Kosaraju 1982, Leroux 2009. But not even primitive recursive bound known.

Superset reachability

Problem: Given a CRN, states x and y, figure out whether some $z \geq y$ is reachable from x.
Can be done in $2^{O(n \log(n))}$ space, where n = [number of species] + log(x) + log(y). This is nearly optimal.

Rackoff, "The Covering and Boundedness Problems For Vector Addition Systems" (1978)

Better characterize non-uniform computation

Deterministic, halting

s-space bounded TM can be simulated with log(s) reactions [???]

Cardoza, Lipton, Meyer, "Exponential Space Complete Problems for Petri Nets and Commutative Semigroups" (1976)

Almost optimal: To figure out whether Y or N can be produced for log(s) reactions can be done in $s^{O(\log(\log(s))}$ space.

Rackoff, "The Covering and Boundedness Problems For Vector Addition Systems" (1978)

Characterize in terms of time-complexity, non-uniform circuit families, etc [???]

Deterministic, stabilizing

???

Fuel and energy use

Tagged CRNs: explicit sources of mass and energy

s-space-bounded computation can be computed by a logically-reversible tagged CRN using poly(s) molecular count

$$A \longleftrightarrow A + B \qquad \longleftrightarrow \qquad \mathit{Fuel1} + A \longleftrightarrow A + B + \mathit{Fuel2}$$

Condon, Manuch, Thachuk "Less haste, less waste: on recycling and its limits in strand displacement systems" (2012)

Thachuk, "Space and energy efficient molecular programming", PhD thesis (2012)

New discoveries about the thermodynamics of computation?

Models of cellular regulatory networks

Relevant insights for systems/synthetic biology?

I would argue that CRNs are a good programming language for strand displacement cascades.

Amounts persist unless explicitly consumed or produced (passive information storage in amount)

Digital stoichiometries

But in cells: ???

Everything is consumed and turned-over (active information storage)

Most regulation is catalytic

Saturating rate laws: eg Hill-functions

Acknowledgements

UCSF Center for Systems & Synth Bio
Winfree group (Caltech)
Seelig group (UW)

Luca Cardelli
Ho-Lin Chen
Yuan-Jyue Chen
Anne Condon
Matthew Cook
Rachel Cummings

David Doty
Lulu Qian
Georg Seelig
Niranjan Srinivas
Chris Thachuk
Erik Winfree

Workshop on Programming CRNs

BIRS

NSF

NSF MPP grant

COMPUTING RESEARCH ASSOCIATION CRA

COMPUTING COMMUNITY CONSORTIUM CCC

CI Fellows

UCSF
center for systems
& synthetic biology
an NIGMS national systems biology center

Real programmers code in CHEMISTRY