Computation by (not about) chemistry

Workshop on mathematical trends in reaction network theory
University of Copenhagen, July 2015

David Doty
The software of life

How does the cell compute?
The software of life

How does the cell compute?

chemistry / geometry
The software of life

How does the cell compute?

What is possible to compute with chemistry or geometry?
Chemical reaction networks (CRN)
Chemical reaction networks (CRN)

\[ R \rightarrow P_1 + P_2 \]
Chemical reaction networks (CRN)

\[ R \rightarrow P_1 + P_2 \]

\[ A + B \rightarrow C \]
Chemical reaction networks (CRN)

\[ R \rightarrow P_1 + P_2 \]

\[ A + B \rightarrow C \]

\[ X + Y \rightarrow X + Z \]
Chemical reaction networks (CRN)

\[ R \rightarrow P_1 + P_2 \]
\[ A + B \rightarrow C \]
\[ X + Y \rightarrow X + Z \]
\[ A + Z \rightarrow \text{(anonymous waste product)} \]
Chemical reaction networks (CRN)

\[ R \rightarrow P_1 + P_2 \]

\[ A + B \rightarrow C \]

\[ X + Y \rightarrow X + Z \]

\[ A + Z \rightarrow \]

\[ (\text{anonymous waste product}) \]

\[ X \rightarrow 2X \]

\[ (\text{anonymous fuel source}) \]
Chemical reaction networks (CRN)

\[ R^{2.5} \rightarrow P_1 + P_2 \]

\[ A + B \rightarrow^1 C \]

\[ X + Y \rightarrow^5 X + Z \]

\[ A + Z \rightarrow^{0.1} \]

(anonymous waste product)

\[ X \rightarrow^{0.1} 2X \]

(anonymous fuel source)
What behavior is possible for chemistry in principle?
What behavior is possible for chemistry in principle?

found in biology

inspiration
What behavior is possible for chemistry in principle?

formally definable CRNs

found in biology

this talk

inspiration
What behavior is possible for chemistry in principle?

- formally definable CRNs
- actual chemicals
- found in biology

this talk
ultimate interest
inspiration
Can we compute with chemistry?

“Not every crazy CRN you scribble on paper describes actual chemicals!”
Can we compute with chemistry?

“Not every crazy CRN you scribble on paper describes actual chemicals!”

Response to objection: Soloveichik et al. [PNAS 2010] showed a physical implementation of every CRN, using DNA strand displacement
Can we compute with chemistry?

“Not every crazy CRN you scribble on paper describes actual chemicals!”

Response to objection: Soloveichik et al. [PNAS 2010] showed a physical implementation of every CRN, using DNA strand displacement

\[ X_1 + X_2 \rightarrow X_3 \]
Can we compute with chemistry?

“Not every crazy CRN you scribble on paper describes actual chemicals!”

Response to objection: Soloveichik et al. [PNAS 2010] showed a physical implementation of every CRN, using DNA strand displacement

\[ X_1 + X_2 \rightarrow X_3 \]
Can we compute with chemistry?

“Not every crazy CRN you scribble on paper describes actual chemicals!”

Response to objection: Soloveichik et al. [PNAS 2010] showed a physical implementation of every CRN, using DNA strand displacement

\[ X_1 + X_2 \rightarrow X_3 \]
Can we compute with chemistry?

“Not every crazy CRN you scribble on paper describes actual chemicals!”

Response to objection: Soloveichik et al. [PNAS 2010] showed a physical implementation of every CRN, using DNA strand displacement

\[ X_1 + X_2 \rightarrow X_3 \]
Why compute with chemistry?

versus
Why compute with chemistry? versus speed?
Why compute with chemistry?

slower versus faster

speed?
Why compute with chemistry?

slower versus faster

speed?
Why compute with chemistry?

- slower
- 
- versus
- 
- faster
- component size?

× speed?
Why compute with chemistry?

- Slower: \( \approx 10-100 \text{ nm} \)
- Faster: component size?

versus

- Speed?
Why compute with chemistry?

versus

slower

≈ 10-100 nm

component size?

faster

≈ 10-100 nm
Why compute with chemistry?

slower

≈ 10-100 nm

versus

speed?

component size?

X

faster

≈ 10-100 nm
Why compute with chemistry?

- Slower versus faster
- Component size?
- Biocompatible with biological or other "wet environments"?

- Cells: "smart drug" released only in certain cellular conditions
- Bioreactors: "chemical controller" to optimize yield of metabolically produced biofuels/drugs/etc.
Why compute with chemistry?

- Slower vs. faster:
  - Slower: ≈ 10-100 nm
  - Faster: ≈ 10-100 nm

- Component size:
  - Yes: Compatible with biological or other “wet environments”?

- Not easily:
  - Yes: “smart drug” released only in certain cellular conditions
  - Not easily: “chemical controller” to optimize yield of metabolically produced biofuels/drugs/etc.

Douglas et al, Science 2012
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

Boolean logic
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

- Boolean logic
- Signal processing
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

- Boolean logic
- Oscillation
- Signal processing
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

- **Boolean logic**
- **discrete algorithms**
- **analog computing**
- **oscillation**
- **signal processing**
Integer-valued kinetic CRN model
Integer-valued kinetic CRN model

- **species**: \( \{X, Y, \ldots\} \)
Integer-valued kinetic CRN model

- **species**: \{X, Y, ...\}

- **reactions**:
  \[ X \xrightarrow{k_1} W + 2Y + Z \]
  \[ A + B \xrightarrow{k_2} X \]
Integer-valued kinetic CRN model

- **species**: \{X, Y, \ldots\}
- **state**: integer vector of *counts* $s = (#X, #Y, \ldots)$
- **reactions**:
  - $X \xrightarrow{k_1} W + 2Y + Z$
  - $A + B \xrightarrow{k_2} X$
Integer-valued kinetic CRN model

- **species**: \{X, Y, \ldots\}
- **state**: integer vector of counts $s = (\#X, \#Y, \ldots)$
- **reactions**:
  - $X \xrightarrow{k_1} W + 2Y + Z$  \hspace{1cm} $k_1 \cdot \#X$
  - $A + B \xrightarrow{k_2} X$  \hspace{1cm} $k_2 \cdot \#A \cdot \#B / \text{volume}$
Integer-valued kinetic CRN model

- **species**: \{X, Y, \ldots\}
- **reactions**:
  - \( X \xrightarrow{k_1} W + 2Y + Z \)  
    \( k_1 \cdot \#X \)
  - \( A + B \xrightarrow{k_2} X \)  
    \( k_2 \cdot \#A \cdot \#B \text{ / volume} \)
- **state**: integer vector of *counts* \( s = (\#X, \#Y, \ldots) \)
- **rate** of reaction:
  - Prob[some reaction] = \[ \frac{\text{rate of that reaction}}{\text{sum of all reaction rates}} \]
Integer-valued kinetic CRN model

- **species**: \( \{ X, Y, \ldots \} \)
- **reactions**:
  - \( X \xrightarrow{k_1} W + 2Y + Z \)
  - \( A + B \xrightarrow{k_2} X \)
- **state**: integer vector of counts \( s = (\#X, \#Y, \ldots) \)
- **rate of reaction**:
  - \( k_1 \cdot \#X \)
  - \( k_2 \cdot \#A \cdot \#B / \text{volume} \)

\[
\text{Prob}[\text{some reaction}] = \frac{\text{rate of that reaction}}{\text{sum of all reaction rates}}
\]

\[
\text{E[time until next reaction]} = \frac{1}{\text{rate}}
\]
CRN function computation (example)

function: \( f(x) = x/2 \)
CRN function computation (example)

function: \( f(x) = x/2 \)
input species: \( X \)
output species: \( Y \)
initial state: \( \{x \in X, 0 \in Y\} \)
CRN function computation (example)

function: $f(x) = x/2$

input species: $X$

output species: $Y$

initial state: $\{x \ X, \ 0 \ Y\}$

reactions: $X \xleftrightarrow{1} Y$
CRN function computation (example)

function: \( f(x) = \frac{x}{2} \)  

input species: \( X \)  
output species: \( Y \)  
initial state: \( \{x X, 0 Y\} \)

reactions: \( X \xleftrightarrow{\frac{1}{1}} Y \)

\#Y = \frac{x}{2} \text{ expected at equilibrium}
CRN function computation (example)

function: \( f(x) = \frac{x}{2} \)

input species: \( X \)

output species: \( Y \)

initial state: \( \{x \ X, 0 \ Y\} \)

reactions:

\( X \xrightleftharpoons{1} Y \)

\#Y = \frac{x}{2} \) expected at equilibrium

reactions:

\( X \xrightarrow{1} Y \)

\( X \xrightarrow{1} \)

\#Y stabilizes, with expected value \( \frac{x}{2} \)
CRN function computation (example)

function: \( f(x) = \frac{x}{2} \)

input species: \( X \)

output species: \( Y \)

initial state: \( \{x \ X, \ 0 \ Y\} \)

reactions: \( X \xleftrightarrow{\ 2 \ } Y \)

\#Y = \frac{x}{2} \) expected at equilibrium

reactions: \( X \xrightarrow{\ 1 \ } Y \)

\( X \xrightarrow{\ 1 \ } \)

\#Y stabilizes, with expected value \( \frac{x}{2} \) and \( \frac{x}{3} \)
Rate-independent CRN computation

What can CRNs compute when we don't know/can't control the rates?
Rate-independent CRN computation

What can CRNs compute when we don't know/can't control the rates?
Rate-independent CRN computation

What can CRNs compute when we don't know/can't control the rates?
Rate-independent CRN computation (a.k.a. “stable”, “deterministic”)
Rate-independent CRN computation (a.k.a. “stable”, “deterministic”)

not the mass-action model!!
CRN function computation (example)

function: $f(x) = 2x$
CRN function computation (example)

function: \( f(x) = 2x \)      
input species: \( X \)      
output species: \( Y \)
CRN function computation (example)

function: \( f(x) = 2x \)

input species: \( X \)
output species: \( Y \)

reactions: ??
CRN function computation (example)

function: \( f(x) = 2x \)

input species: \( X \)

output species: \( Y \)

reactions: \( X \rightarrow 2Y \)
CRN function computation (example)

function: \( f(x) = 2x \)

input species: \( X \)
output species: \( Y \)

reactions: \( X \rightarrow 2Y \)
CRN function computation (example)

function: \( f(x) = 2x \)

input species: \( X \)

output species: \( Y \)

reactions: \( X \rightarrow 2Y \)
CRN function computation (example)

function: $f(x) = 2x$

input species: $X$

output species: $Y$

reactions: $X \rightarrow 2Y$
function: $f(x) = x/2$
CRN function computation (example)

**function:** \( f(x) = x/2 \)

**reactions:** \( 2X \rightarrow Y \)
CRN function computation (example)

function: \( f(x) = \frac{x}{2} \)

reactions: \( 2X \rightarrow Y \)
CRN function computation (example)

function: \( f(x) = \frac{x}{2} \)

reactions: \( 2X \rightarrow Y \)
CRN function computation (example)

**function**: \( f(x) = x/2 \)

**reactions**: \( 2X \rightarrow Y \)
CRN function computation (example)

function: \( f(x_1, x_2) = x_1 + x_2 \)
CRN function computation (example)

**function:** \( f(x_1, x_2) = x_1 + x_2 \)

**reactions:**
\[
\begin{align*}
X_1 & \rightarrow Y \\
X_2 & \rightarrow Y
\end{align*}
\]
CRN function computation (example)

function: \( f(x_1, x_2) = x_1 + x_2 \)

reactions:  \( X_1 \rightarrow Y \)  
\( X_2 \rightarrow Y \)
CRN function computation (example)

function: \( f(x_1, x_2) = x_1 - x_2 \)
CRN function computation (example)

function: \( f(x_1, x_2) = x_1 - x_2 \)

reactions: \( X_1 \rightarrow Y \)
\( X_2 + Y \rightarrow \)
CRN function computation (example)

function: \( f(x_1, x_2) = x_1 - x_2 \)

reactions:

- \( X_1 \rightarrow Y \)
- \( X_2 + Y \rightarrow \)
CRN function computation (example)

**function:** \( f(x_1, x_2) = x_1 - x_2 \)

**reactions:**

\[
\begin{align*}
X_1 & \rightarrow Y \\
X_2 + Y & \rightarrow
\end{align*}
\]
CRN function computation (example)

function: \( f(x_1, x_2) = x_1 - x_2 \)

reactions:
- \( X_1 \rightarrow Y \)
- \( X_2 + Y \rightarrow \)
CRN function computation (example)

**function:** \( f(x_1, x_2) = x_1 - x_2 \)

**reactions:**

- \( X_1 \rightarrow Y \)
- \( X_2 + Y \rightarrow \)
CRN function computation (example)

function: \( f(x_1, x_2) = \min\{x_1, x_2\} \)
CRN function computation (example)

function: \( f(x_1, x_2) = \min\{x_1, x_2\} \)

reactions: \( X_1 + X_2 \rightarrow Y \)
CRN function computation (example)

function: \( f(x_1, x_2) = \min\{x_1, x_2\} \)

reactions: \( X_1 + X_2 \rightarrow Y \)
CRN function computation (example)

function: \( f(x_1, x_2) = \min\{x_1, x_2\} \)

reactions: \( X_1 + X_2 \rightarrow Y \)
CRN function computation (example)

function: \( f(x_1, x_2) = \max\{x_1, x_2\} \)
CRN function computation (example)

**function**: \[ f(x_1, x_2) = \max\{x_1, x_2\} = x_1 + x_2 - \min\{x_1, x_2\} \]
CRN function computation (example)

function: \( f(x_1, x_2) = \max\{x_1, x_2\} = x_1 + x_2 - \min\{x_1, x_2\} \)

reactions: 
\[ X_1 \rightarrow Y + X_1' \]
\[ X_2 \rightarrow Y + X_2' \]
CRN function computation (example)

function: \( f(x_1, x_2) = \max\{x_1, x_2\} = x_1 + x_2 - \min\{x_1, x_2\} \)

reactions: 
\[
\begin{align*}
X_1 &\rightarrow Y + X'_1 \\
X_2 &\rightarrow Y + X'_2
\end{align*}
\]
CRN function computation (example)

function: \( f(x_1, x_2) = \max\{x_1, x_2\} = x_1 + x_2 - \min\{x_1, x_2\} \)

reactions: 
- \( X_1 \rightarrow Y + X_1' \)
- \( X_2 \rightarrow Y + X_2' \)
- \( X_1' + X_2' \rightarrow K \)
CRN function computation (example)

**function**: \( f(x_1, x_2) = \max\{x_1, x_2\} = x_1 + x_2 - \min\{x_1, x_2\} \)

**reactions**: 
- \( X_1 \rightarrow Y + X_1' \)
- \( X_2 \rightarrow Y + X_2' \)
- \( X_1' + X_2' \rightarrow K \)
CRN function computation (example)

**function**: \( f(x_1, x_2) = \max\{x_1, x_2\} = x_1 + x_2 - \min\{x_1, x_2\} \)

**reactions**: 
- \( X_1 \rightarrow Y + X'_1 \)
- \( X_2 \rightarrow Y + X'_2 \)
- \( X'_1 + X'_2 \rightarrow K \)
- \( K + Y \rightarrow \)
CRN function computation (example)

function: \( f(x_1, x_2) = \max\{x_1, x_2\} = x_1 + x_2 - \min\{x_1, x_2\} \)

reactions:
\[ X_1 \rightarrow Y + X_1' \]
\[ X_2 \rightarrow Y + X_2' \]
\[ X_1' + X_2' \rightarrow K \]
\[ K + Y \rightarrow \]
CRN predicate computation (example)

**Predicate**: \( p(x) \): parity of \( x \) ("yes" \( \Leftrightarrow \) \( x \) is odd)
CRN predicate computation (example)

**Predicate:** $p(x)$: parity of $x$ ("yes" $\iff x$ is odd)

**Initial state:** \{ $x \ X$, 1 $N$, 0 $Y$ \}
CRN predicate computation (example)

**Predicate**: \( p(x) \): parity of \( x \) ("yes" \( \iff \) \( x \) is odd)

**Initial state**: \( \{ x \ X , \ 1 \ N , \ 0 \ Y \} \)

**Reactions**:
- \( N + X \rightarrow Y \)
- \( Y + X \rightarrow N \)
CRN predicate computation (example)

**Predicate:** \( p(x) \): parity of \( x \) ("yes" \( \Leftrightarrow \) \( x \) is odd)

**Initial state:** \( \{ x \ X , 1 \ N , 0 \ Y \} \)

**Reactions:**
- \( N + X \rightarrow Y \)
- \( Y + X \rightarrow N \)
**CRN predicate computation (example)**

**Predicate:** $p(x)$: parity of $x$ ("yes" $\iff x$ is odd)

**Initial state:** \{ $x$, $X$, 1 $N$, 0 $Y$ \}

**Reactions:**
- $N + X \rightarrow Y$
- $Y + X \rightarrow N$
CRN predicate computation (example)

**predicate**: $p(x)$: parity of $x$ ("yes" $\iff x$ is odd)

**initial state**: $\{ x X, 1 N, 0 Y \}$

**reactions**: 
- $N + X \rightarrow Y$
- $Y + X \rightarrow N$
CRN predicate computation (example)

**Predicate:** \( p(x) \): parity of \( x \) ("yes" \( \iff \) \( x \) is odd)

**Initial state:** \( \{ x \ X \ , \ 1 \ N \ , \ 0 \ Y \} \)

**Reactions:**
\[
N + X \rightarrow Y \quad X \quad N \\
Y + X \rightarrow N
\]
CRN predicate computation (example)

**predicate**: $p(x)$: parity of $x$ ("yes" $\iff x$ is odd)

**initial state**: \{ $x$ $X$, 1 $N$, 0 $Y$ \}

**reactions**: $N + X \rightarrow Y$

$Y + X \rightarrow N$
CRN predicate computation (example)

**predicate**: $p(x)$: parity of $x$ ("yes" $\Leftrightarrow x$ is odd)

**initial state**: \{ $x \ X$, 1 $N$, 0 $Y$ \}

**reactions**: $N + X \rightarrow Y$

$Y + X \rightarrow N$
CRN predicate computation (example)

**Predicate:** $p(x_1, x_2): "x_1 > x_2"$?
CRN predicate computation (example)

**predicate**: $p(x_1, x_2): \ "x_1 > x_2\ "$?

**initial state**: $\{ x_1 X_1, x_2 X_2, 1 N \}$
CRN predicate computation (example)

**predicate:** $p(x_1, x_2): \text{"}x_1 > x_2\text{"}$?

**initial state:** $\{ x_1 X_1, x_2 X_2, 1 N \}$

**reactions:**
- $N + X_1 \rightarrow Y$
- $Y + X_2 \rightarrow N$
CRN predicate computation (example)

predicate: \( p(x_1, x_2): \text{"}x_1 > x_2\text{"}\)?

initial state: \{ \( x_1 \ X_1 \), \( x_2 \ X_2 \), 1 \( N \) \}

reactions: \( N + X_1 \rightarrow Y \)
\( Y + X_2 \rightarrow N \)
CRN predicate computation (example)

**predicate:** $p(x_1, x_2): \text{"}x_1 > x_2\text{"}?$

**initial state:** \{ $x_1 X_1$, $x_2 X_2$, 1 $N$ \}

**reactions:**
- $N + X_1 \rightarrow Y$
- $Y + X_2 \rightarrow N$
CRN predicate computation (example)

**Predicate:** \( p(x_1, x_2): \text{“}x_1 > x_2\text{”} \)?

**Initial state:** \( \{ x_1 X_1, x_2 X_2, 1 N \} \)

**Reactions:**

\( N + X_1 \rightarrow Y \)

\( Y + X_2 \rightarrow N \)
CRN predicate computation (example)

predicate: \( p(x_1, x_2): "x_1 > x_2" \) ?

initial state: \( \{ x_1 X_1, x_2 X_2, 1 N \} \)

reactions: 
\[ N + X_1 \rightarrow Y \]
\[ Y + X_2 \rightarrow N \]
CRN predicate computation (example)

**Predicate:** \( p(x_1, x_2): \text{"}x_1 > x_2\text{"} \)?

**Initial state:** \( \{ x_1 \ X_1 , x_2 \ X_2 , 1 \ N \} \)

**Reactions:**
- \( N + X_1 \rightarrow Y \)
- \( Y + X_2 \rightarrow N \)
CRN predicate computation (example)

predicate: \( p(x_1, x_2): \text{"}x_1 > x_2\text{"} \)?

initial state: \( \{ x_1 X_1, x_2 X_2, 1 N \} \)

reactions: \( N + X_1 \rightarrow Y \)
\( Y + X_2 \rightarrow N \)
CRN predicate computation (example)

**predicate:** $p(x_1, x_2)$: “$x_1 > x_2$”?

**initial state:** $\{x_1 X_1, x_2 X_2, 1 N\}$

**reactions:**

- $N + X_1 \rightarrow Y$
- $Y + X_2 \rightarrow N$
CRN predicate computation (example)

**predicate:** $p(x_1, x_2): \text{“}x_1 > x_2\text{”}$?

**initial state:** $\{x_1 X_1, x_2 X_2, 1 N\}$

**reactions:**

- $N + X_1 \rightarrow Y$
- $Y + X_2 \rightarrow N$
CRN predicate computation (example)

predicate: $p(x_1, x_2)$: “$x_1 = x_2$”? 
CRN predicate computation (example)

**predicate**: \( p(x_1, x_2): "x_1 = x_2" \)?

**initial state**: \( \{ x_1 X_1, x_2 X_2, 1 Y \} \)
CRN predicate computation (example)

**predicate**: \( p(x_1, x_2) \): “\( x_1 = x_2 \)”?

**initial state**: \( \{ x_1 X_1, x_2 X_2, 1 Y \} \)
CRN predicate computation (example)

predicate: $p(x_1, x_2): \ "x_1 = x_2\ "$?

initial state: $\{ x_1 X_1, x_2 X_2, 1 Y \}$

reactions:

$X_1 + X_2 \rightarrow Y$
$Y + N \rightarrow Y$
$X_1 + Y \rightarrow X_1 + N$
$X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**predicate:** \( p(x_1, x_2): "x_1 = x_2" \)?

**initial state:** \{ \( x_1 X_1 \), \( x_2 X_2 \), 1 \( Y \) \}

**reactions:**

\[
\begin{align*}
X_1 + X_2 & \rightarrow Y \\
Y + N & \rightarrow Y \\
X_1 + Y & \rightarrow X_1 + N \\
X_2 + Y & \rightarrow X_2 + N
\end{align*}
\]
CRN predicate computation (example)

**predicate:** $p(x_1, x_2): \text{“}x_1 = x_2\text{”}?$

**initial state:** $\{x_1 X_1, x_2 X_2, 1 Y\}$

**reactions:**

$X_1 + X_2 \rightarrow Y$

$Y + N \rightarrow Y$

$X_1 + Y \rightarrow X_1 + N$

$X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**predicate:** \( p(x_1, x_2): \text{“} x_1 = x_2 \text{”} \)?

**initial state:** \( \{ x_1 X_1, x_2 X_2, 1 Y \} \)

**reactions:**
- \( X_1 + X_2 \rightarrow Y \)
- \( Y + N \rightarrow Y \)
- \( X_1 + Y \rightarrow X_1 + N \)
- \( X_2 + Y \rightarrow X_2 + N \)
CRN predicate computation (example)

Predicate: \( p(x_1, x_2): \text{"} x_1 = x_2 \text{"} \)?

Initial state: \{ \( x_1 X_1, x_2 X_2, 1 Y \) \}

Reactions:

\[
\begin{align*}
X_1 + X_2 &\rightarrow Y \\
Y + N &\rightarrow Y \\
X_1 + Y &\rightarrow X_1 + N \\
X_2 + Y &\rightarrow X_2 + N
\end{align*}
\]
CRN predicate computation (example)

**predicate:** \( p(x_1, x_2): \) “\( x_1 = x_2 \)”?

**initial state:** \{ \( x_1 X_1, x_2 X_2, 1 Y \) \}

**reactions:**
\[
\begin{align*}
X_1 + X_2 & \rightarrow Y \\
Y + N & \rightarrow Y \\
X_1 + Y & \rightarrow X_1 + N \\
X_2 + Y & \rightarrow X_2 + N
\end{align*}
\]
CRN predicate computation (example)

**Predicate:** \( p(x_1, x_2): \text{"}x_1 = x_2\text{"} \) ?

**Initial state:** \( \{ x_1 X_1, x_2 X_2, 1 Y \} \)

**Reactions:**
- \( X_1 + X_2 \rightarrow Y \)
- \( Y + N \rightarrow Y \)
- \( X_1 + Y \rightarrow X_1 + N \)
- \( X_2 + Y \rightarrow X_2 + N \)
CRN predicate computation (example)

**predicate**: $p(x_1, x_2)$: “$x_1 = x_2$”?

**initial state**: \{ $x_1$ $X_1$, $x_2$ $X_2$, 1 $Y$ \}

**reactions**:

- $X_1 + X_2 \rightarrow Y$
- $Y + N \rightarrow Y$
- $X_1 + Y \rightarrow X_1 + N$
- $X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**Predicate:** \( p(x_1, x_2): \text{“}x_1 = x_2\text{”)? \)

**Initial state:** \{ \( x_1 X_1 \), \( x_2 X_2 \), 1 \( Y \) \}

**Reactions:**
- \( X_1 + X_2 \rightarrow Y \)
- \( Y + N \rightarrow Y \)
- \( X_1 + Y \rightarrow X_1 + N \)
- \( X_2 + Y \rightarrow X_2 + N \)
CRN predicate computation (example)

**Predicate:** $p(x_1, x_2): "x_1 = x_2"$?

**Initial state:** $\{ x_1 X_1, x_2 X_2, 1 Y \}$

**Reactions:**
- $X_1 + X_2 \rightarrow Y$
- $Y + N \rightarrow Y$
- $X_1 + Y \rightarrow X_1 + N$
- $X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**predicate:** \( p(x_1, x_2) \): “\( x_1 = x_2 \)”?

**initial state:** \( \{ x_1 \ X_1, x_2 \ X_2, 1 \ Y \} \)

**reactions:**
- \( X_1 + X_2 \rightarrow Y \)
- \( Y + N \rightarrow Y \)
- \( X_1 + Y \rightarrow X_1 + N \)
- \( X_2 + Y \rightarrow X_2 + N \)
CRN predicate computation (example)

**predicate:** $p(x_1,x_2): "x_1 = x_2"$?

**initial state:** $\{ x_1 X_1 , x_2 X_2 , 1 Y \}$

**reactions:**
- $X_1 + X_2 \rightarrow Y$
- $Y + N \rightarrow Y$
- $X_1 + Y \rightarrow X_1 + N$
- $X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**predicate:** \( p(x_1, x_2): \text{“} x_1 = x_2 \text{”} \)?

**initial state:** \{ \( x_1 X_1 \), \( x_2 X_2 \), \( 1 Y \) \}

**reactions:**
- \( X_1 + X_2 \rightarrow Y \)
- \( Y + N \rightarrow Y \)
- \( X_1 + Y \rightarrow X_1 + N \)
- \( X_2 + Y \rightarrow X_2 + N \)
CRN predicate computation (example)

**Predicate:** $p(x_1, x_2): \text{“} x_1 = x_2 \text{”}?$

**Initial state:** $\{ x_1 X_1, x_2 X_2, 1 Y \}$

**Reactions:**

- $X_1 + X_2 \rightarrow Y$
- $Y + N \rightarrow Y$
- $X_1 + Y \rightarrow X_1 + N$
- $X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

predicate: $p(x_1, x_2): "x_1 = x_2"$?

initial state: $\{ x_1 X_1, x_2 X_2, 1 Y \}$

reactions:

- $X_1 + X_2 \rightarrow Y$
- $Y + N \rightarrow Y$
- $X_1 + Y \rightarrow X_1 + N$
- $X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**Predicate:** \( p(x_1, x_2): \text{“}x_1 = x_2\text{”}\)?

**Initial State:** \{ \( x_1 X_1 \), \( x_2 X_2 \), 1 \( Y \) \}

**Reactions:**
- \( X_1 + X_2 \rightarrow Y \)
- \( Y + N \rightarrow Y \)
- \( X_1 + Y \rightarrow X_1 + N \)
- \( X_2 + Y \rightarrow X_2 + N \)
CRN predicate computation (example)

**predicate:** $p(x_1, x_2): \ "x_1 = x_2\"$?

**initial state:** \{ $x_1 X_1, x_2 X_2, 1 Y$ \}

**reactions:**

- $X_1 + X_2 \rightarrow Y$
- $Y + N \rightarrow Y$
- $X_1 + Y \rightarrow X_1 + N$
- $X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**predicate:** $p(x_1, x_2): \ "x_1 = x_2\ "$?

**initial state:** $\{ x_1 X_1 , x_2 X_2 , 1 Y \}$

**reactions:**

- $X_1 + X_2 \rightarrow Y$
- $Y + N \rightarrow Y$
- $X_1 + Y \rightarrow X_1 + N$
- $X_2 + Y \rightarrow X_2 + N$
CRN predicate computation (example)

**predicate**: $p(x_1, x_2): \text{"}x_1 = x_2\text{"}$?

**initial state**: \{ $x_1 X_1$, $x_2 X_2$, 1 $Y$ \}

**reactions**: $X_1 + X_2 \rightarrow Y$

\[\begin{align*}
Y + N \rightarrow Y \\
X_1 + Y \rightarrow X_1 + N \\
X_2 + Y \rightarrow X_2 + N
\end{align*}\]
CRN predicate computation (example)

**predicate**: \( p(x_1, x_2): \ "3x_1 > x_2/2" \)?

**initial state**: \{ \( x_1 X_1 \), \( x_2 X_2 \), \( 1 N \) \}
CRN predicate computation (example)

**Predicate:** $p(x_1, x_2): \text{“}3x_1 > x_2/2\text{”}\ ?$

**Initial state:** $\{ x_1 X_1 , x_2 X_2 , 1 N \}$

**Reactions:** $X_1 \rightarrow 3Z_1$
CRN predicate computation (example)

**predicate:** $p(x_1, x_2): \ "3x_1 > \frac{x_2}{2}\$?

**initial state:** $\{ x_1 \ X_1 \ , \ x_2 \ X_2 \ , \ 1 \ N \}$

**reactions:**
- $X_1 \rightarrow 3Z_1$
- $2X_2 \rightarrow Z_2$
CRN predicate computation (example)

predicate: \( p(x_1, x_2): \text{“}3x_1 \geq x_2/2\text{”} \)?

initial state: \( \{ x_1 X_1, x_2 X_2, 1 N \} \)

reactions:
\[
X_1 \rightarrow 3Z_1 \\
2X_2 \rightarrow Z_2 \\
N + Z_1 \rightarrow Y \\
Y + Z_2 \rightarrow N
\]
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{$yes,no$\}$, $x_1,\ldots,x_k \in \mathbb{N}$
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{\text{yes}, \text{no}\}$, $x_1,\ldots,x_k \in \mathbb{N}$

**initial state:** $#X_1 = x_1,\ldots,#X_k = x_k$, constant counts of other species
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{\text{yes, no}\}, \quad x_1,\ldots,x_k \in \mathbb{N}$

**initial state:** $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{\text{yes, no}\}, \quad x_1,\ldots,x_k \in \mathbb{N}$

**initial state:** $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output $\varphi(s)$ of state $s$:** consensus vote (if voters unanimous)
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{\text{yes}, \text{no}\}, \ x_1,\ldots,x_k \in \mathbb{N}$

**initial state:** $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output $\varphi(s)$ of state $s$:** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{\text{yes, no}\}$, $x_1,\ldots,x_k \in \mathbb{N}$

**initial state:** $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output $\varphi(s)$ of state $s$:** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output

**stable computation:** for all states $s$ reachable from the initial state $x$, a correct output-stable state $o$ is reachable from $s$
Stable predicate computation (definition)

**task:** compute \( p(x_1,\ldots,x_k) \in \{\text{yes, no}\}, \quad x_1,\ldots,x_k \in \mathbb{N} \)

**initial state:** \( \#X_1 = x_1,\ldots,\#X_k = x_k \), constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output \( \varphi(s) \) of state \( s \):** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output

**stable computation:** for all states \( s \) reachable from the initial state \( x \), a correct output-stable state \( o \) is reachable from \( s \)
Stable predicate computation (definition)

**task:** compute \( p(x_1,\ldots,x_k) \in \{\text{yes, no}\}, \quad x_1,\ldots,x_k \in \mathbb{N} \)

**initial state:** \( \#X_1 = x_1,\ldots,\#X_k = x_k \), constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output \( \varphi(s) \) of state \( s \):** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output

**stable computation:** for all states \( s \) reachable from the initial state \( x \), a correct output-stable state \( o \) is reachable from \( s \)
Stable predicate computation (definition)

**task**: compute $p(x_1,\ldots,x_k) \in \{\text{yes}, \text{no}\}, \ x_1,\ldots,x_k \in \mathbb{N}$

**initial state**: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

**votes**: two disjoint subsets of species: “yes” and “no” voters

**output $\varphi(s)$ of state $s$**: consensus vote (if voters unanimous)

**output-stable state**: all states reachable from it have same output

**stable computation**: for all states $s$ reachable from the initial state $x$, a correct output-stable state $o$ is reachable from $s$

\[
\varphi(o) = p(x_1,\ldots,x_k)
\]
Stable predicate computation (definition)

**task:** compute \( p(x_1,\ldots,x_k) \in \{\text{yes},\text{no}\}, \quad x_1,\ldots,x_k \in \mathbb{N} \)

**initial state:** \( \#X_1 = x_1,\ldots,\#X_k = x_k \), constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output \( \phi(s) \) of state \( s \):** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output

**stable computation:** for all states \( s \) reachable from the initial state \( x \), a correct output-stable state \( o \) is reachable from \( s \)

\[ \phi(o) = p(x_1,\ldots,x_k) \]
Stable predicate computation (definition)

**task:** compute \( p(x_1, ..., x_k) \in \{\text{yes}, \text{no}\}, \ x_1, ..., x_k \in \mathbb{N} \)

**initial state:** \( \#X_1 = x_1, ..., \#X_k = x_k \), constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output \( \varphi(s) \) of state \( s \):** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output

**stable computation:** for all states \( s \) reachable from the initial state \( x \), a correct output-stable state \( o \) is reachable from \( s \)

\[ \varphi(o) = p(x_1, ..., x_k) \]
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{\text{yes},\text{no}\}, \quad x_1,\ldots,x_k \in \mathbb{N}$

**initial state:** $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output $\varphi(s)$ of state $s$:** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output

**stable computation:** for all states $s$ reachable from the initial state $x$, a correct output-stable state $o$ is reachable from $s$

\[
\varphi(o) = p(x_1,\ldots,x_k)
\]
Stable predicate computation (definition)

**task:** compute $p(x_1,\ldots,x_k) \in \{\text{yes, no}\}$, $x_1,\ldots,x_k \in \mathbb{N}$

**initial state:** \(\#X_1 = x_1,\ldots,\#X_k = x_k\), constant counts of other species

**votes:** two disjoint subsets of species: “yes” and “no” voters

**output \(\varphi(s)\) of state \(s\):** consensus vote (if voters unanimous)

**output-stable state:** all states reachable from it have same output

**stable computation:** for all states \(s\) reachable from the initial state \(x\), a correct output-stable state \(o\) is reachable from \(s\)

\[ \varphi(o) = p(x_1,\ldots,x_k) = \varphi(o_2) \]
Stable predicate computation (definition)

**Task:** compute $p(x_1, \ldots, x_k) \in \{\text{yes, no}\}$, $x_1, \ldots, x_k \in \mathbb{N}$

**Initial state:** $\#X_1 = x_1, \ldots, \#X_k = x_k$, constant counts of other species

**Votes:** two disjoint subsets of species: “yes” and “no” voters

**Output $\phi(s)$ of state $s$:** consensus vote (if voters unanimous)

**Output-stable state:** all states reachable from it have same output

**Stable computation:** for all states $s$ reachable from the initial state $x$, a correct output-stable state $o$ is reachable from $s$

$\phi(o) = p(x_1, \ldots, x_k) = \phi(o_2)$
Stable predicate computation (definition)

**task**: compute \( p(x_1, \ldots, x_k) \in \{\text{yes, no}\}, \quad x_1, \ldots, x_k \in \mathbb{N} \)

**initial state**: \( \#X_1 = x_1, \ldots, \#X_k = x_k \), constant counts of other species

**votes**: two disjoint subsets of species: “yes” and “no” voters

**output \( \varphi(s) \) of state \( s \)**: consensus vote (if voters unanimous)

**output-stable state**: all states reachable from it have same output

**stable computation**: for all states \( s \) reachable from the initial state \( x \), a correct output-stable state \( o \) is reachable from \( s \)

\[
\varphi(o) = p(x_1, \ldots, x_k) = \varphi(o_2)
\]
Stable predicate computation (definition)

function \( p(x_1,\ldots,x_k) \in \{\text{yes, no}\}, \quad x_1,\ldots,x_k \in \mathbb{N} \)

task: compute \( p(x_1,\ldots,x_k) \)

initial state: \( \#X_1 = x_1,\ldots,\#X_k = x_k \), constant counts of other species

votes: two disjoint subsets of species: “yes” and “no” voters

output \( \varphi(s) \) of state \( s \): consensus vote (if voters unanimous)

output-stable state: all states reachable from it have same output

stable computation: for all states \( s \) reachable from the initial state \( x \), a correct output-stable state \( o \) is reachable from \( s \)

\[ \varphi(o) = p(x_1,\ldots,x_k) = \varphi(o_2) \]
Stable predicate computation (definition)

**Function**

**Task:** compute \( p(x_1, ..., x_k) \in \{\text{yes}, \text{no}\}, \quad x_1, ..., x_k \in \mathbb{N} \)

**Initial state:** \( \#X_1 = x_1, ..., \#X_k = x_k \), constant counts of other species

**Votes:** two disjoint subsets of species: “yes” and “no” voters

**Output \( \varphi(s) \) of state \( s \):** consensus vote (if voters unanimous) \( \#Y \)

**Output-stable state:** all states reachable from it have same output

**Stable computation:** for all states \( s \) reachable from the initial state \( x \), a correct output-stable state \( o \) is reachable from \( s \)

\[
\varphi(o) = p(x_1, ..., x_k) = \varphi(o_2)
\]
Stable computation characterization

**Theorem:** A predicate/function is stably computed by a CRN if and only if it is semilinear.

[Chen, D, Soloveichik, *DNA Computing* 2012]
**Theorem**: A predicate/function is stably computed by a CRN if and only if it is *semilinear*.

**semilinear predicate** = Boolean combination of *threshold* and *mod* tests

[Chen, D, Soloveichik, *DNA Computing* 2012]
Stable computation characterization

**Theorem**: A predicate/function is stably computed by a CRN if and only if it is *semilinear*.

semilinear predicate = Boolean combination of *threshold* and *mod* tests

\[ x_1 - 3x_2 < -7 \]

[Chen, D, Soloveichik, *DNA Computing* 2012]
Stable computation characterization

**Theorem**: A predicate/function is stably computed by a CRN if and only if it is *semilinear*.

semilinear predicate = Boolean combination of threshold and mod tests

\[ 2x_1 + x_2 \equiv 3 \mod 5 \quad x_1 - 3x_2 < -7 \]

[Chen, D, Soloveichik, *DNA Computing* 2012]
Theorem: A predicate/function is stably computed by a CRN if and only if it is semilinear.

semilinear predicate = Boolean combination of threshold and mod tests

\[ 2x_1 + x_2 \equiv 3 \pmod{5} \quad x_1 - 3x_2 < -7 \]

semilinear function ≈ piecewise linear

[Angluin, Aspnes, Eisenstat, Principles of Distributed Computing 2006]
[Chen, D, Soloveichik, DNA Computing 2012]
Real-valued CRNs

Theorem from previous slide: A function is stably computed by a integer-valued CRN if and only if it is semilinear.
Real-valued CRNs

Theorem from previous slide: A function is stably computed by a integer-valued CRN if and only if it is semilinear.

Real-valued version: A function is stably computed by a real-valued CRN if and only if it is continuous and piecewise linear.

[Chen, D, Soloveichik, Innovations in Theoretical Computer Science 2014]
Real-valued CRNs

Theorem from previous slide: A function is stably computed by a integer-valued CRN if and only if it is \textit{semilinear}.

$\approx$ piecewise linear functions with “discontinuous” pieces

Real-valued version: A function is stably computed by a real-valued CRN if and only if it is \textit{continuous} and \textit{piecewise linear}.

[Chen, D, Soloveichik, Innovations in Theoretical Computer Science 2014]
Time complexity of stable computation

$$n = \# \text{ molecules in initial state} \approx \text{ volume}$$
Time complexity of stable computation

\[ n = \# \text{ molecules in initial state} \approx \text{volume} \]

**Theorem:** \(O(n)\) if initial state contains only input molecules

[Angluin, Aspnes, Eisenstat, *Principles of Distributed Computing* 2006 (predicates)]
[D, Hajiaghayi, *DNA Computing* 2013 (functions)]
Time complexity of stable computation

\[ n = \# \text{molecules in initial state} \approx \text{volume} \]

**Theorem**: \( O(n) \) if initial state contains only input molecules

[Angluin, Aspnes, Eisenstat, *Principles of Distributed Computing* 2006 (predicates)]
[D, Hajiaghayi, *DNA Computing* 2013 (functions)]

**Theorem**: \( O(\log^5 n) \) otherwise (if CRN can start with a **leader**: single copy of a certain non-input molecule)

[Angluin, Aspnes, Eisenstat, *Symposium on Distributed Computing* 2006 (predicates)]
[Chen, D, Soloveichik, *DNA Computing* 2012 (functions)]
Time complexity of stable computation

\[ n = \# \text{ molecules in initial state} \approx \text{volume} \]

**Theorem:** \( O(n) \) if initial state contains only input molecules

[Angluin, Aspnes, Eisenstat, *Principles of Distributed Computing* 2006 (predicates)]
[D, Hajiaghayi, *DNA Computing* 2013 (functions)]

**Theorem:** \( O(\log^5 n) \) otherwise (if CRN can start with a **leader:** single copy of a certain non-input molecule)

[Angluin, Aspnes, Eisenstat, *Symposium on Distributed Computing* 2006 (predicates)]
[Chen, D, Soloveichik, *DNA Computing* 2012 (functions)]

Polylogarithmic time = “fast” = polynomial in binary expansion of \( n \)

Linear time = “slow” = exponential in binary expansion of \( n \)
Time complexity in CRNs

time until next reaction = exponential random variable

<table>
<thead>
<tr>
<th>reaction</th>
<th>expected time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \rightarrow W + 2Y + Z$</td>
<td>$1 / #X$</td>
</tr>
<tr>
<td>$A + B \rightarrow X$</td>
<td>volume / ($#A \cdot #B$)</td>
</tr>
</tbody>
</table>
Time complexity (example)

\[ \{n \times\} \]

\[ X \rightarrow Y + Y \]
Time complexity (example)

\[ \{n \ X\} \]

\[ X \rightarrow Y + Y \]

E[time to consume all X] =
Time complexity (example)

\[
\{n \ X\} \\
X \rightarrow Y + Y
\]

\[E[\text{time to consume all } X] = E[\text{time to consume } 1^{\text{st}} X] + E[\text{time to consume } 2^{\text{nd}} X] + E[\text{time to consume } 3^{\text{rd}} X] + \ldots + E[\text{time to consume final } X]\]
Time complexity (example)

\[ \{n \times X\} \]
\[ X \rightarrow Y + Y \]

\[
E[\text{time to consume all } X] = \quad E[\text{time to consume } 1^{\text{st}} X] \quad \frac{1}{n} \\
+ E[\text{time to consume } 2^{\text{nd}} X] \quad \frac{1}{(n-1)} \\
+ E[\text{time to consume } 3^{\text{rd}} X] \quad \frac{1}{(n-2)} \\
+ \ldots \\
+ E[\text{time to consume final } X] \quad \frac{1}{1}
\]
Time complexity (example)

\[ \{n \ X\} \]
\[ X \rightarrow Y + Y \]

\[
E[\text{time to consume all X}] = E[\text{time to consume 1}\text{st X}] + E[\text{time to consume 2}\text{nd X}] + E[\text{time to consume 3}\text{rd X}] + \ldots + E[\text{time to consume final X}] \approx \log n
\]

\[
= \frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \ldots + \frac{1}{1}
\]

\[= \frac{1/1}{n} + \frac{1/1}{n-1} + \frac{1/1}{n-2} + \ldots + \frac{1/1}{1} \approx \log n
\]
Time complexity (example)

\{n \times\}, \text{ volume } n
X + X \rightarrow Y
Time complexity (example)

\[ \{n \ X\}, \text{ volume } n \]
\[ X + X \rightarrow Y \]

“finite density constraint”
Time complexity (example)

$$\{n \times X\}, \text{ volume } n$$

$$X + X \rightarrow Y$$

E[time to consume all $X$] = $n/n^2 + n/(n-2)^2 + n/(n-4)^2 + \ldots + n$

“finite density constraint”
Time complexity (example)

\$\{n \times \}, \text{ volume } n \$

\$X + X \rightarrow Y\$

volume \#X^2

"finite density constraint"

\[ E[\text{time to consume all } X] = \frac{n}{n^2} + \frac{n}{(n-2)^2} + \frac{n}{(n-4)^2} + \ldots + n \]

\[ < n(\frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{6^2} + \frac{1}{8^2} + \ldots) \]
Time complexity (example)

\{n \ X\}, \text{ volume } n

\begin{align*}
X + X & \rightarrow Y \\
\text{volume} & \quad \#X^2 \\
\end{align*}

“finite density constraint”

\[
E[\text{time to consume all } X] = \frac{n}{n^2} + \frac{n}{(n-2)^2} + \frac{n}{(n-4)^2} + \ldots + n
\]

\[
< n\left(\frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{6^2} + \frac{1}{8^2} + \ldots\right)
\]

= \mathcal{O}(n)
Time complexity (example)

\( \{n \ X\}, \ \text{volume} \ n \)

\( X + X \rightarrow Y \)

volume \#X^2

\[ \text{E[time to consume all } X] = \frac{n}{n^2} + \frac{n}{(n-2)^2} + \frac{n}{(n-4)^2} + \ldots + n \]

\(< n(1/2^2 + 1/4^2 + 1/6^2 + 1/8^2 + \ldots) \]

\[ = O(n) = \Theta(n) \]

“finite density constraint”
Time complexity (example)

\{n \, X\}, \text{ volume } n

\[ X + X \rightarrow Y \]

volume \quad \#X^2 \quad \text{“finite density constraint”}

\[ \text{E[time to consume all } X\text{]} = \frac{n}{n^2} + \frac{n}{(n-2)^2} + \frac{n}{(n-4)^2} + \ldots + n \]

< \[ n\left(\frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{6^2} + \frac{1}{8^2} + \ldots\right) \]

= \[ O(n) = \Theta(n) \]

Is there a faster CRN for dividing by 2 from initial state \{n \, X\}? 

\text{(Open problem)}
Time complexity (leader election)

\{n \ L\}, \text{volume } n

\ L + \ L \rightarrow L
Time complexity (leader election)

\{n \ L\}, \text{ volume } n \ \ L + L \rightarrow L \ \ \ \ \ \ \ \ \ E[\text{time get to } 1 \ L] = O(n)
Time complexity (leader election)

\{n \, L\}, \text{ volume } n

\begin{align*}
L + L &\rightarrow L \\
E[\text{time get to 1 } L] &= O(n)
\end{align*}

Is there a faster CRN?
Time complexity (leader election)

\[ \{n \ L\} \text{, volume } n \]

\[ L + L \rightarrow L \]

\[ \mathbb{E}[\text{time get to } 1 \ L] = O(n) \]

Is there a faster CRN?

- If we really abuse the CRN model, probably (use \(2X \rightarrow 3X\))
Time complexity (leader election)

\{n \; L\}, \text{ volume } n
\quad L + L \rightarrow L

\text{E[time get to 1 } L) = O(n)

Is there a faster CRN?

- If we really abuse the CRN model, probably (use \(2X \rightarrow 3X\))
- In mass-conserving CRNs, we don't know
Time complexity (leader election)

\[ \{n \ L\}, \ \text{volume } n \]
\[ L + L \rightarrow L \]

\[ \text{E[time get to 1 } L]\text{] = } O(n) \]

Is there a faster CRN?

- If we really abuse the CRN model, probably (use \(2X \rightarrow 3X\))
- In mass-conserving CRNs, we don't know
  - [Angluin, Aspnes, Eisenstat, Symposium on Distributed Computing, 2006] show a CRN that seems to work in simulation, with small probability of error (unproven)
Time complexity (leader election)

\[
\{n \ L\}, \ \text{volume} \ n \\
L + L \rightarrow L
\]

\[E[\text{time get to 1 } L] = O(n)\]

Is there a faster CRN?

- If we really abuse the CRN model, probably (use \(2X \rightarrow 3X\))
- In mass-conserving CRNs, we don't know
  - [Angluin, Aspnes, Eisenstat, Symposium on Distributed Computing 2006] show a CRN that seems to work in simulation, with small probability of error (unproven)
  - [Alistarh and Gelashvili, International Colloquium on Automata, Languages, and Programming 2015] show a CRN with \(\log^3 n\) species that provably elects a leader in \(\log^3 n\) time
Time complexity (leader election)

\{n L\}, volume \ n
L + L \rightarrow L

E[time get to 1 L] = O(n)

Is there a faster CRN?

• If we really abuse the CRN model, probably (use 2X → 3X)
• In mass-conserving CRNs, we don't know
  − [Angluin, Aspnes, Eisenstat, *Symposium on Distributed Computing* 2006] show a CRN that seems to work in simulation, with small probability of error (unproven)
  − [Alistarh and Gelashvili, *International Colloquium on Automata, Languages, and Programming* 2015] show a CRN with \( \log^3 n \) species that provably elects a leader in \( \log^3 n \) time
  − if we require 0 probability of error, no [D and Soloveichik, *in submission*]
What if we allow a small probability of error? (rate-dependent CRN computation)
CRNs with small probability of error are Turing universal

[Angluin, Aspnes, Eisenstat, *Symposium on Distributed Computing* 2006]
[Soloveichik, Cook, Winfree, Bruck, *Natural Computing* 2008]
CRNs with small probability of error are Turing universal

(Informally) A CRN can simulate any algorithm, with a small chance of error.

[Angluin, Aspnes, Eisenstat, *Symposium on Distributed Computing* 2006]
[Soloveichik, Cook, Winfree, Bruck, *Natural Computing* 2008]
CRNs with small probability of error are Turing universal

(Informally) A CRN can simulate any algorithm, with a small chance of error.

Implication: General CRN long-term behavior cannot be predicted faster than by simulating.

[Angluin, Aspnes, Eisenstat, *Symposium on Distributed Computing* 2006]
[Soloveichik, Cook, Winfree, Bruck, *Natural Computing* 2008]
Counter (register) machine
Counter (register) machine
Counter (register) machine

“input” counter

r → s → t
Counter (register) machine

1) \textit{dec}(r)
2) \textit{inc}(s)
3) \textit{inc}(s)
4) \textit{inc}(s)
5) \textit{dec}(t)
6) \textit{inc}(s)
Counter (register) machine

1) \text{dec}(r)
2) \text{inc}(s)
3) \text{inc}(s)
4) \text{inc}(s)
5) \text{dec}(t)
6) \text{inc}(s)

“input” counter

\begin{itemize}
\item [r]
\item [s]
\item [t]
\end{itemize}
Counter (register) machine

1) $dec(r)$
2) $inc(s)$
3) $inc(s)$
4) $inc(s)$
5) $dec(t)$
6) $inc(s)$
Counter (register) machine

1) $\text{dec}(r)$
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$
6) $\text{inc}(s)$

“input” counter
Counter (register) machine

1) $\text{dec}(r)$
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$
6) $\text{inc}(s)$
Counter (register) machine

1) \textit{dec}(r)
2) \textit{inc}(s)
3) \textit{inc}(s)
4) \textit{inc}(s)
5) \textit{dec}(t)
6) \textit{inc}(s)
Counter (register) machine

1) $\text{dec}(r)$
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$
6) $\text{inc}(s)$
Counter (register) machine

1) \textit{dec}(r) \quad \text{if empty goto 6}
2) \textit{inc}(s)
3) \textit{inc}(s)
4) \textit{inc}(s)
5) \textit{dec}(t) \quad \text{if empty goto 1}
6) \textit{inc}(s)
Counter (register) machine

1) \textit{dec}(r) \quad \text{if empty goto 6}
2) \textit{inc}(s)
3) \textit{inc}(s)
4) \textit{inc}(s)
5) \textit{dec}(t) \quad \text{if empty goto 1}
6) \textit{inc}(s)
Counter (register) machine

1) $dec(r)$  if empty goto 6
2) $inc(s)$
3) $inc(s)$
4) $inc(s)$
5) $dec(t)$  if empty goto 1
6) $inc(s)$
Counter (register) machine

1) $\text{dec}(r)$ if empty goto 6
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$ if empty goto 1
6) $\text{inc}(s)$
Counter (register) machine

1) $\text{dec}(r)$ if empty goto 6
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$ if empty goto 1
6) $\text{inc}(s)$

“input” counter
Counter (register) machine

1) \( \text{dec}(r) \) if empty goto 6
2) \( \text{inc}(s) \)
3) \( \text{inc}(s) \)
4) \( \text{inc}(s) \)
5) \( \text{dec}(t) \) if empty goto 1
6) \( \text{inc}(s) \)
Counter (register) machine

1) $\text{dec}(r)$ if empty goto 6
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$ if empty goto 1
6) $\text{inc}(s)$

“input” counter
Counter (register) machine

1) $\text{dec}(r)$ if empty goto 6
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$ if empty goto 1
6) $\text{inc}(s)$
Counter (register) machine

1) \textit{dec}(r) \quad \text{if empty goto 6}
2) \textit{inc}(s)
3) \textit{inc}(s)
4) \textit{inc}(s)
5) \textit{dec}(t) \quad \text{if empty goto 1}
6) \textit{inc}(s)

“input” counter

\[ \text{r} \quad \text{s} \quad \text{t} \]
Counter (register) machine

1) $\text{dec}(r)$  if empty goto 6
2) $\text{inc}(s)$
3) $\text{inc}(s)$
4) $\text{inc}(s)$
5) $\text{dec}(t)$  if empty goto 1
6) $\text{inc}(s)$

HALT
Counter (register) machine

1) \( \text{dec}(r) \) if empty goto 6
2) \( \text{inc}(s) \)
3) \( \text{inc}(s) \)
4) \( \text{inc}(s) \)
5) \( \text{dec}(t) \) if empty goto 1
6) \( \text{inc}(s) \)

HALT

computes \( f(n) = 3n+1 \)
CRNs can simulate counter machines
CRNs can simulate counter machines

Counter machine:

1. \textit{inc}(r)
2. \textit{dec}(r) if zero goto 1
3. \textit{inc}(s)
4. \textit{dec}(s) if zero goto 2
CRNs can simulate counter machines

Counter machine:
- $r = \text{input } n$, start line 1
  1) $inc(r)$
  2) $dec(r)$ if zero goto 1
  3) $inc(s)$
  4) $dec(s)$ if zero goto 2

CRN:
- initial state \( \{n \ R, \ 1 \ L_1\} \)
CRNs can simulate counter machines

Counter machine:

- \( r = \text{input } n \), start line 1
  1) \( \text{inc}(r) \)
  2) \( \text{dec}(r) \) if zero goto 1
  3) \( \text{inc}(s) \)
  4) \( \text{dec}(s) \) if zero goto 2

CRN:

- initial state \( \{ n \ R, \ 1 \ L_1 \} \)
  \[ L_1 \rightarrow L_2 + R \]
CRNs can simulate counter machines

Counter machine:

\( r = \) input \( n \), start line 1

1) \( inc(r) \)
2) \( dec(r) \) if zero goto 1
3) \( inc(s) \)
4) \( dec(s) \) if zero goto 2

CRN:

initial state \( \{n \ R, \ 1 \ L_1\} \)

\( L_1 \rightarrow L_2 + R \)

\( L_2 + R \rightarrow L_3 \)
CRNs can simulate counter machines

**Counter machine:**
- r = input n, start line 1
  1) $inc(r)$
  2) $dec(r)$ if zero goto 1
  3) $inc(s)$
  4) $dec(s)$ if zero goto 2

**CRN:**
- initial state $\{n R, 1 L_1\}$
  - $L_1 \rightarrow L_2 + R$
  - $L_2 + R \rightarrow L_3$
CRNs can simulate counter machines

Counter machine:

\[ r = \text{input } n, \text{ start line 1} \]

1) \textit{inc}(r)

2) \textit{dec}(r) \textit{if zero goto 1}

3) \textit{inc}(s)

4) \textit{dec}(s) \textit{if zero goto 2}

CRN:

\[ \text{initial state } \{n \ R, \ 1 \ L_1\} \]

\[ L_1 \rightarrow L_2 + R \]

\[ L_2 + R \rightarrow L_3 \text{ ; } L_2 \rightarrow L_1 \]
CRNs can simulate counter machines

Counter machine:
r = input $n$, start line 1

1) $inc(r)$
2) $dec(r)$ if zero goto 1
3) $inc(s)$
4) $dec(s)$ if zero goto 2

CRN:
initial state \{n R, 1 L_1\}

\begin{align*}
L_1 & \rightarrow L_2 + R \\
L_2 + R & \rightarrow L_3 ; L_2 \rightarrow L_1 \\
L_3 & \rightarrow L_4 + S \\
L_4 + S & \rightarrow L_5 ; L_4 \rightarrow L_2
\end{align*}
CRNs can simulate counter machines with probability < 1

Counter machine:

r = input \( n \), start line 1

1) \( \text{inc}(r) \)
2) \( \text{dec}(r) \) if zero goto 1
3) \( \text{inc}(s) \)
4) \( \text{dec}(s) \) if zero goto 2

CRN:

initial state \( \{ n \ R, \ 1 \ L_1 \} \)

\[
\begin{align*}
L_1 & \rightarrow L_2 + R \\
L_2 + R & \rightarrow L_3 \\
L_3 & \rightarrow L_4 + S \\
L_4 + S & \rightarrow L_5 \\
L_2 & \rightarrow L_1 \\
L_4 & \rightarrow L_2
\end{align*}
\]

Need to be very slow!
How to slow down reaction $L_2 \rightarrow L_1$?
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

1 $C_1$, 1 $F$
How to slow down reaction $L_2 \rightarrow L_1$?

Use a **clock**:

1 $C_1$, 1 $F$

\[ F + C_1 \rightarrow F + C_2 \]

\[ F + C_2 \rightarrow F + C_3 \]
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

$1 \ C_1, 1 \ F$, $n \ B$

\[
F + C_1 \rightarrow F + C_2 \quad B + C_2 \rightarrow B + C_1 \\
F + C_2 \rightarrow F + C_3 \quad B + C_3 \rightarrow B + C_2
\]
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

1 $C_1$, 1 $F$, $n$ $B$

\[ F + C_1 \rightarrow F + C_2 \quad B + C_2 \rightarrow B + C_1 \]
\[ F + C_2 \rightarrow F + C_3 \quad B + C_3 \rightarrow B + C_2 \]

reverse-biased random walk
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

1 $C_1$, 1 $F$, $n$ $B$

- $F + C_1 \rightarrow F + C_2$
- $F + C_2 \rightarrow F + C_3$
- $B + C_2 \rightarrow B + C_1$
- $B + C_3 \rightarrow B + C_2$

$C_3$ appears after expected time $\approx n^2$

reverse-biased random walk
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

1 $C_1$, 1 $F$, $n B$

$F + C_1 \rightarrow F + C_2$
$B + C_2 \rightarrow B + C_1$

$F + C_2 \rightarrow F + C_3$
$B + C_3 \rightarrow B + C_2$

reverse-biased random walk

$C_3$ appears after expected time $\approx n^2$
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:
1 $C_1$, 1 $F$, $n B$

$F + C_1 \rightarrow F + C_2$
$F + C_2 \rightarrow F + C_3$
$B + C_2 \rightarrow B + C_1$
$B + C_3 \rightarrow B + C_2$

$C_3 + L_2 \rightarrow C_1 + L_1$

reverse-biased random walk
$C_3$ appears after expected time $\approx n^2$
$E[\text{time for } L_2 + R \rightarrow L_3] \leq n$
Probability 1 computation
Probability 1 computation

- Errr... isn't that stable computation?
Probability 1 computation

- Errr... isn't that stable computation?
- With finite state space, yes.
Probability 1 computation

- Errr... isn't that stable computation?
- With finite state space, yes.

Consider...

\[
\begin{align*}
Y & \overset{1}{\rightarrow} \ \text{initial state } \{1Y,1N\} \\
Y & \overset{2}{\rightarrow} 2Y \\
\end{align*}
\]

stably computes \( \varphi = \text{no} \) (prob < 1)
Probability 1 computation

- Errr... isn't that stable computation?
- With finite state space, yes.

Consider...

\[ Y \xrightarrow{2} 2Y \]
\[ Y \xrightarrow{1} 1 \]
\[ Y \xrightarrow{1} Y \]

computes \( \phi = \text{yes with prob 1} \) (not stably)
Probability 1 computation

- Errr... isn't that stable computation?
- With finite state space, yes.

Consider...

\[
\begin{align*}
Y & \rightarrow 2Y \\
Y & \rightarrow 1 \\
1 & \rightarrow Y
\end{align*}
\]

Theorem: All (Turing) computable predicates/functions can be computed by a CRN with probability 1.

[Cummings, D, Soloveichik, *DNA Computing* 2014]
Acknowledgments

Ho-Lin Chen  Rachel Cummings  David Soloveichik

American Institute of Mathematics