Programming with chemical kinetics

Kinetic networks: From topology to design, Sept 2015

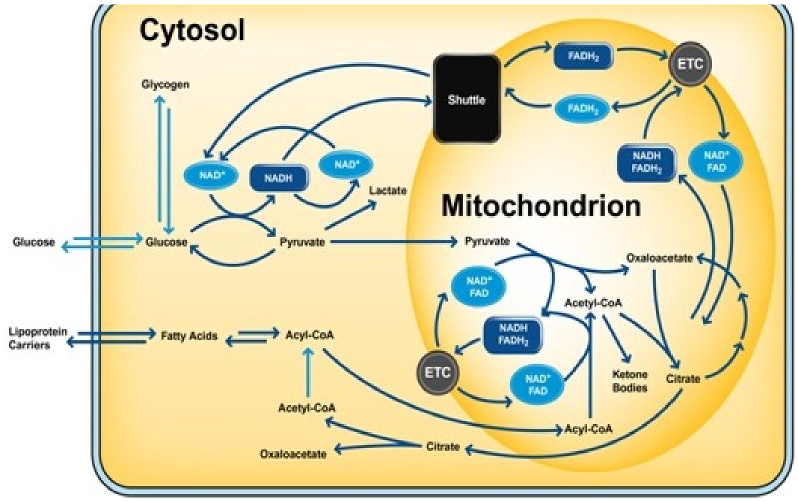
David Doty
Department of Computer Science
University of California, Davis

The software of life

How does the cell compute?

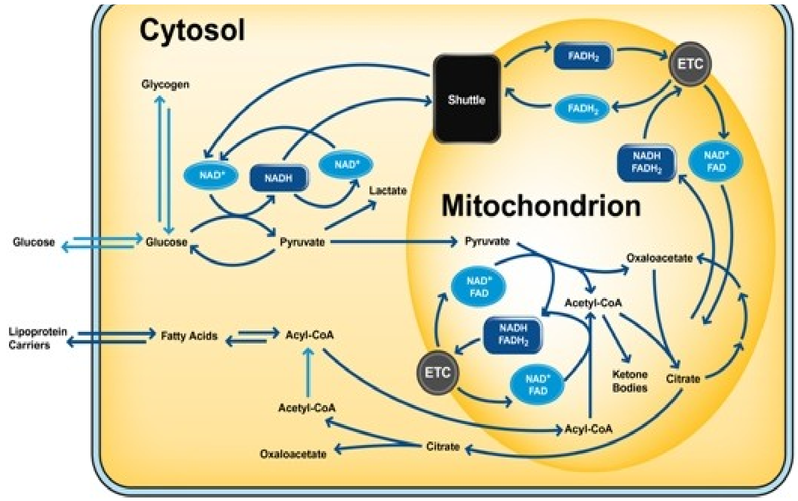
The software of life

How does the cell compute?



chemistry / geometry

The software of life



~~How does the cell compute?~~

What is possible to compute with chemistry?/ ~~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)

$$R \rightarrow P_1 + P_2$$

$$A + B \rightarrow C$$

$$X + Y \rightarrow X + Z$$

$$A + Z \rightarrow$$

(anonymous
waste product)

Chemical reaction networks (CRN)

$$R \rightarrow P_1 + P_2$$

$$A + B \rightarrow C$$

$$X + Y \rightarrow X + Z$$

$$A + Z \rightarrow$$

(anonymous
waste product)

$$X \rightarrow 2X$$

(anonymous
fuel source)

Chemical reaction networks (CRN)

$$R \xrightarrow{2.5} P_1 + P_2$$

$$A + B \xrightarrow{1} C$$

$$X + Y \xrightarrow{5} X + Z$$

$$A + Z \xrightarrow{0.1}$$

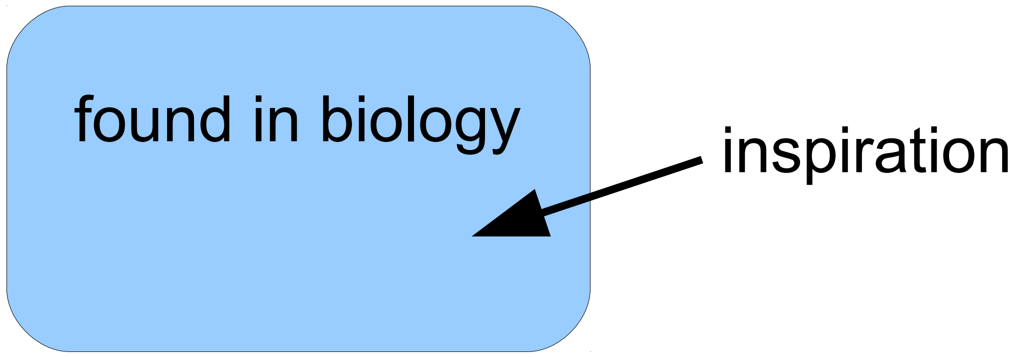
(anonymous
waste product)

$$X \xrightarrow{0.1} 2X$$

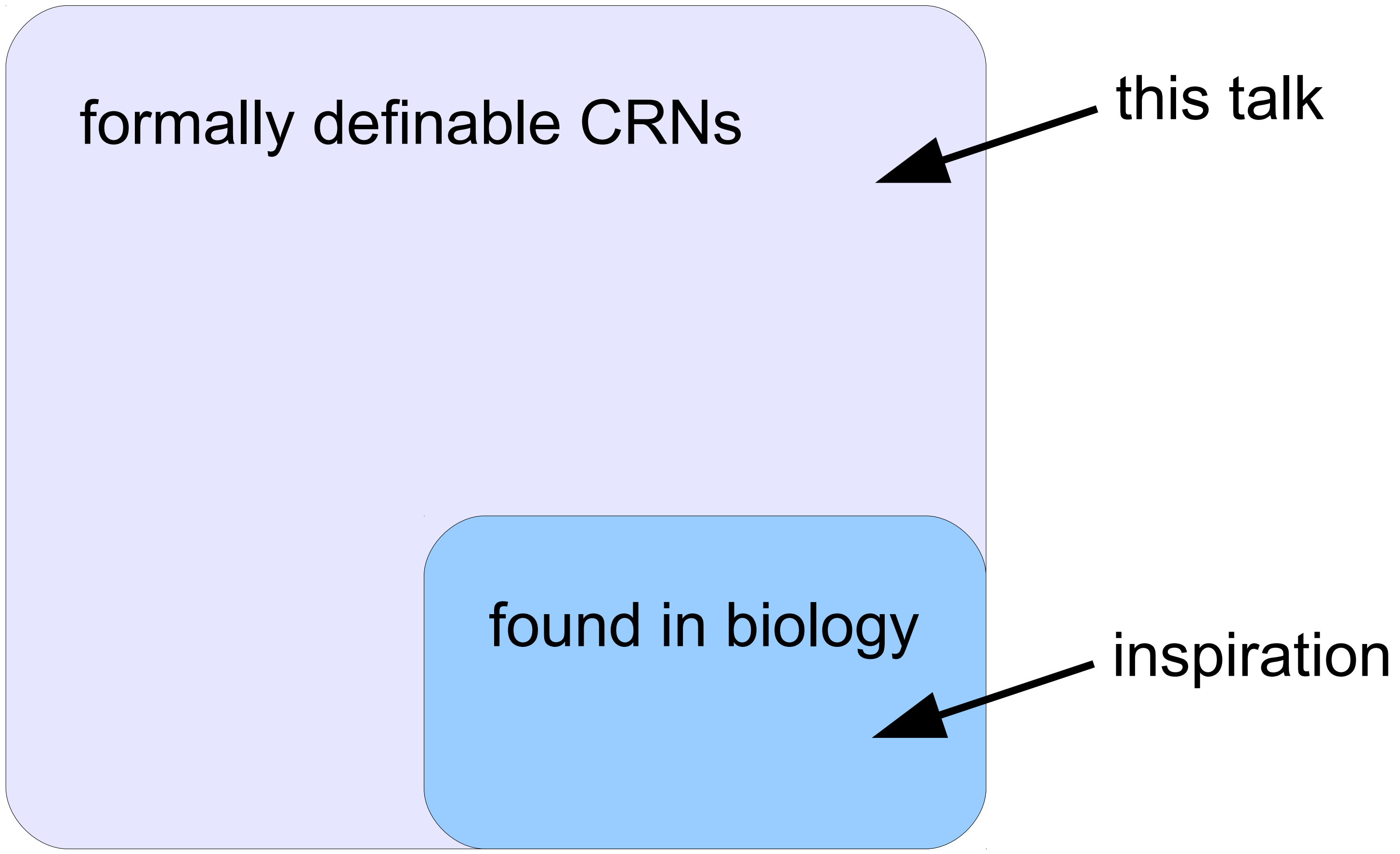
(anonymous
fuel source)

What behavior is possible for chemistry in principle?

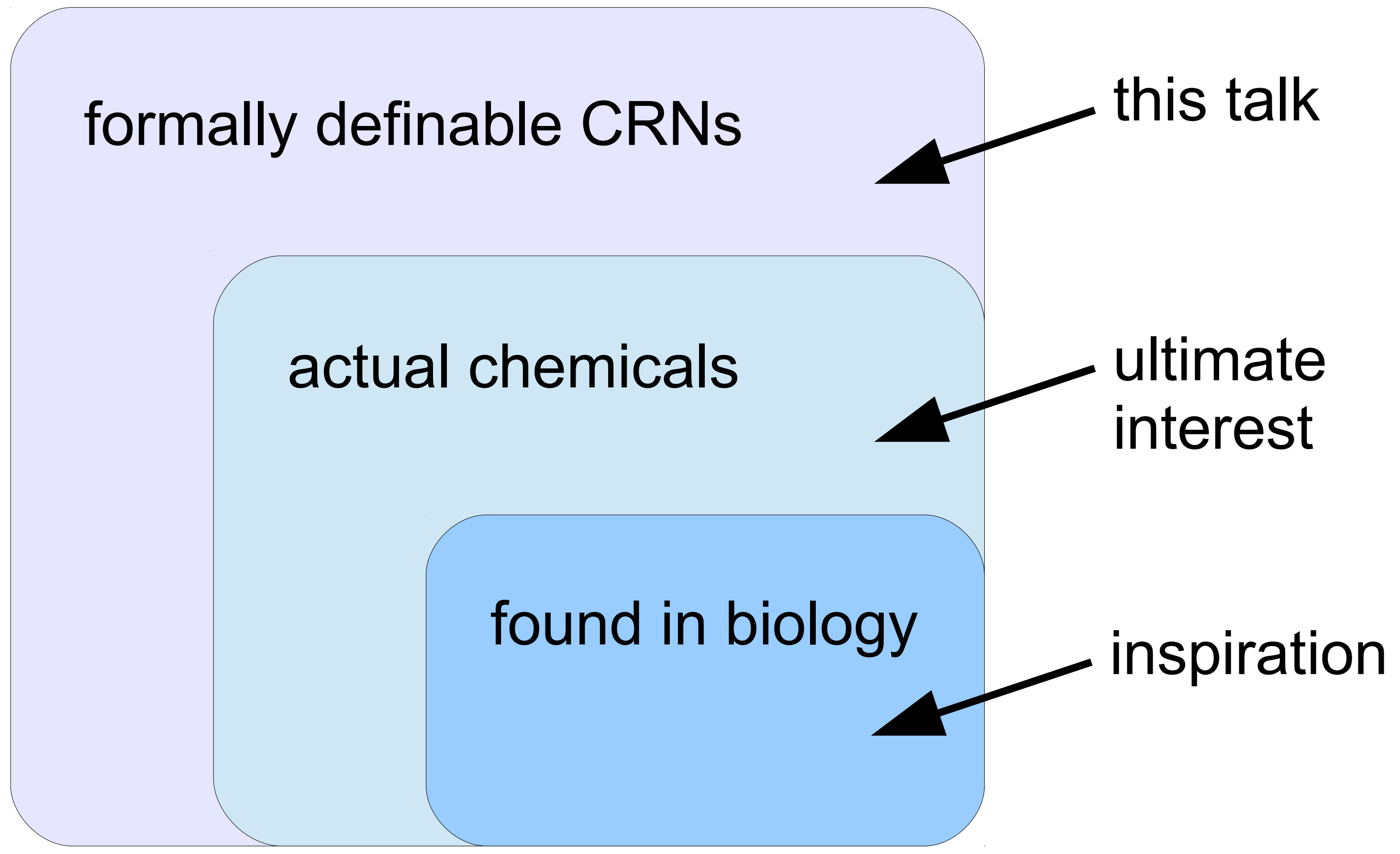
What behavior is possible for chemistry in principle?



What behavior is possible for chemistry in principle?



What behavior is possible for chemistry in principle?



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 <u>every</u> 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 <u>every</u> CRN, using *DNA strand displacement*

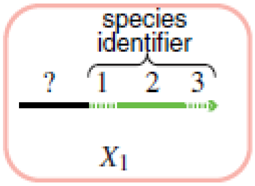
$$X_1 + X_2 \rightarrow X_3$$

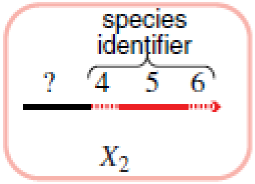
Can we compute with chemistry?

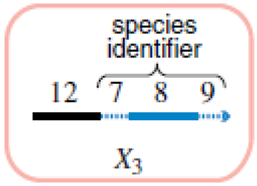
“Not every crazy CRN you scribble on paper describes actual chemicals!”

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

$$X_1 + X_2 \to X_3$$





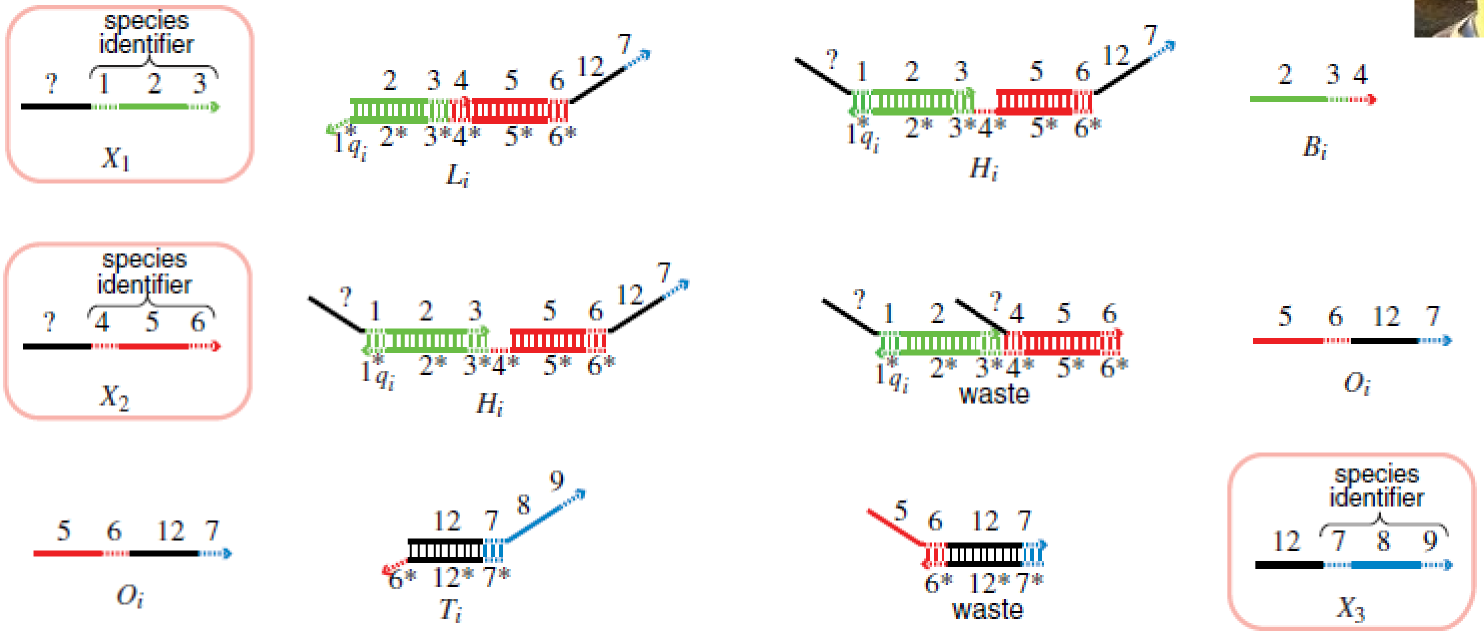


Can we compute with chemistry?

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

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

$$X_1 + X_2 \rightarrow X_3$$

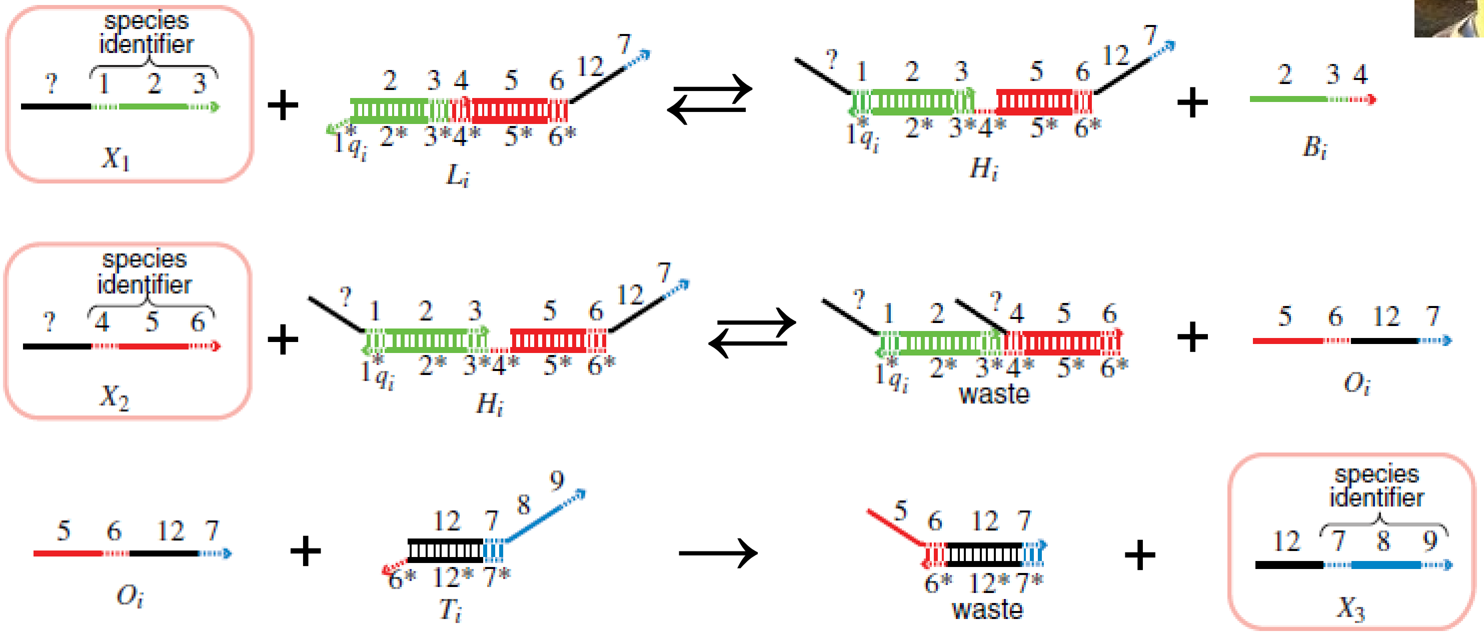


Can we compute with chemistry?

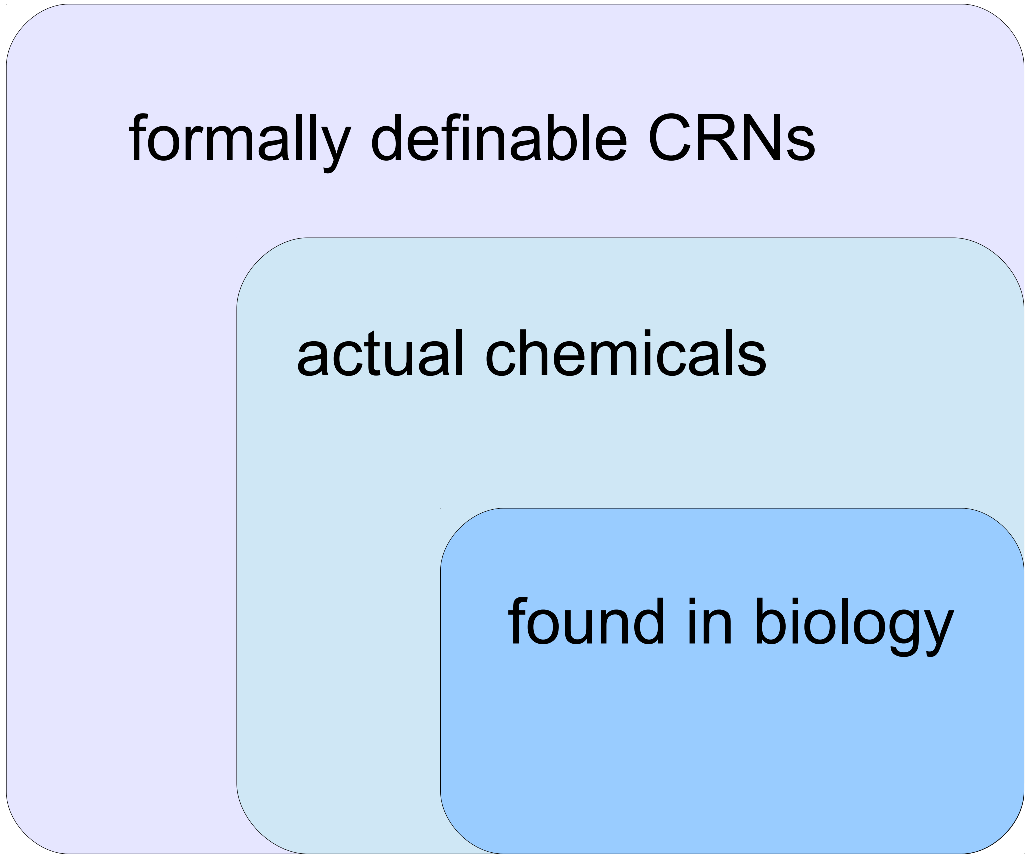
"Not every crazy CRN you scribble on paper describes actual chemicals!"

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

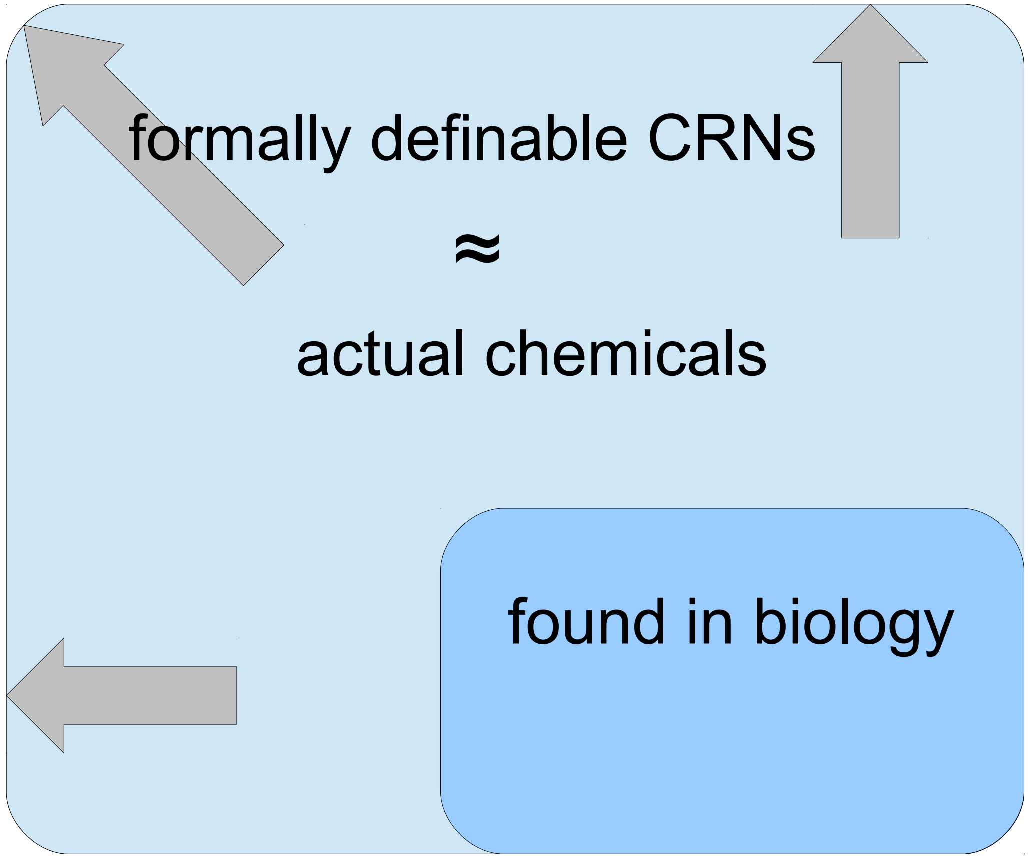
$$X_1 + X_2 \rightarrow X_3$$



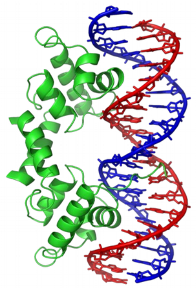
What behavior is possible for chemistry in principle?



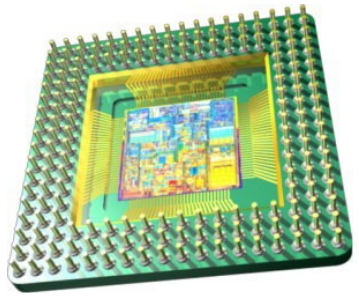
What behavior is possible for chemistry in principle?



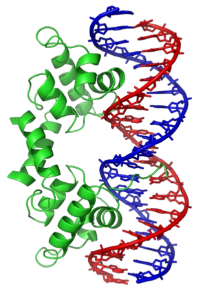
Why compute with chemistry?



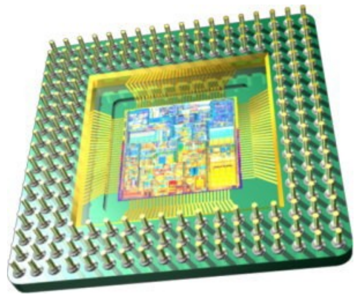
versus



Why compute with chemistry?

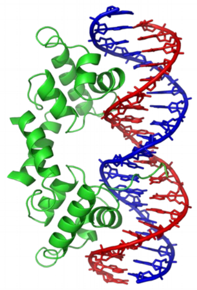


versus

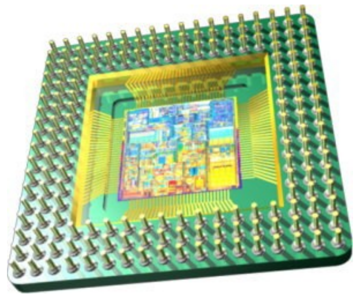


speed?

Why compute with chemistry?



versus

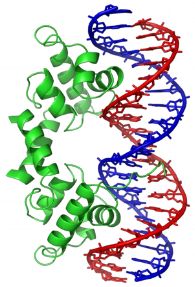


slower

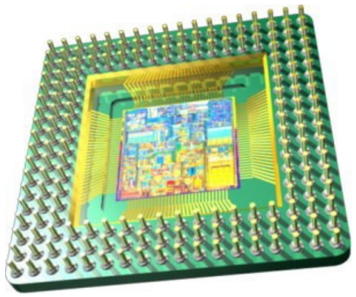
speed?

faster

Why compute with chemistry?



versus

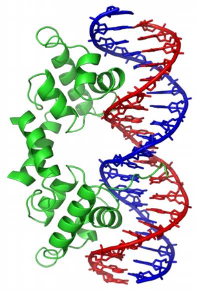


slower

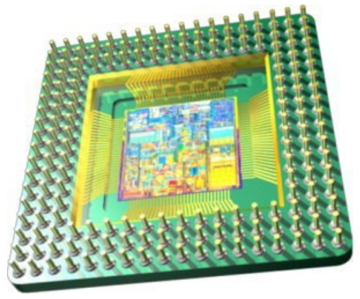
~~speed?~~

faster

Why compute with chemistry?



versus



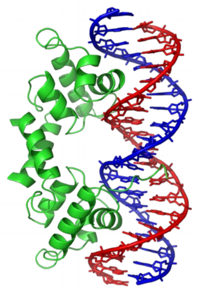
slower

~~speed?~~

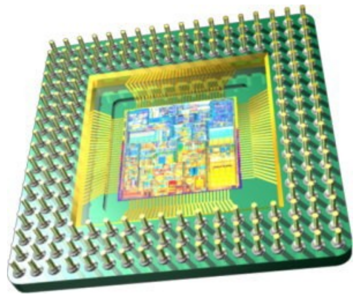
faster

component size?

Why compute with chemistry?

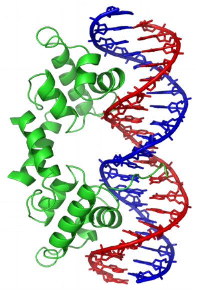


versus

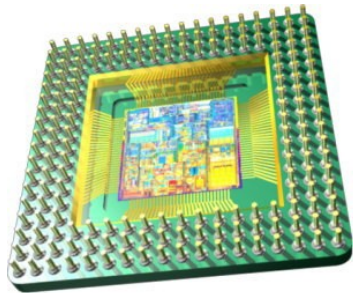


slower	~~speed?~~	faster
≈ 10-100 nm	component size?	

Why compute with chemistry?



versus



slower

~~speed?~~

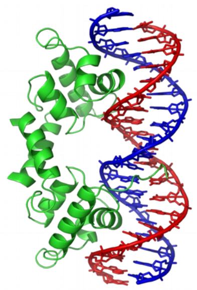
faster

≈ 10-100 nm

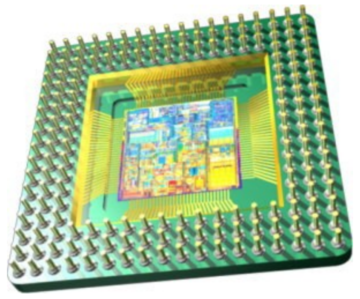
component size?

≈ 10-100 nm

Why compute with chemistry?

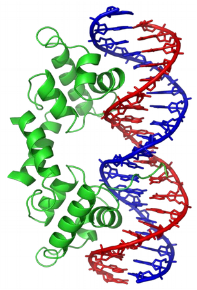


versus

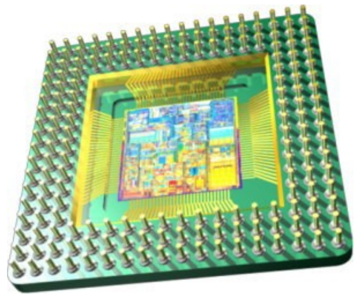


slower	~~speed?~~	faster
≈ 10-100 nm	~~component size?~~	≈ 10-100 nm

Why compute with chemistry?

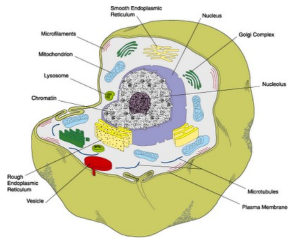


versus



slower	✗ speed?	faster
≈ 10-100 nm	✗ component size?	≈ 10-100 nm
yes	✓ Compatible with biological or other “wet environments”?	not easily

cells

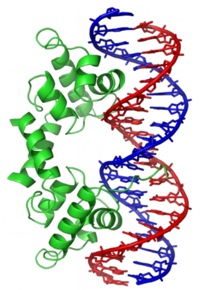


“smart drug” released only in certain cellular conditions

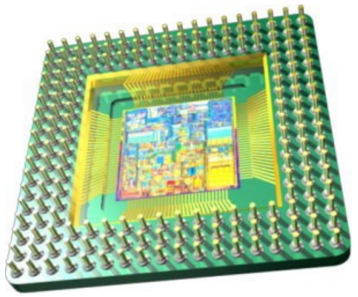
bioreactors

“chemical controller” to optimize yield of metabolically produced biofuels/drugs/etc.

Why compute with chemistry?

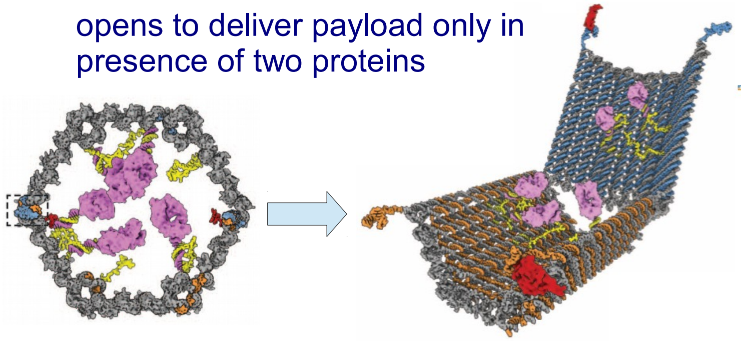


versus



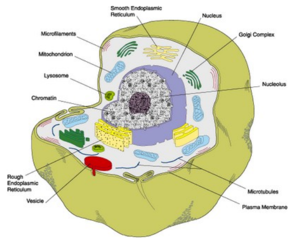
slower	~~speed?~~	faster
≈ 10-100 nm	~~component size?~~	≈ 10-100 nm
yes	✓ Compatible with biological or other “wet environments”?	not easily

opens to deliver payload only in presence of two proteins



Douglas et al, *Science* 2012

cells



“smart drug” released only in certain cellular conditions

bioreactors

“chemical controller” to optimize yield of metabolically produced biofuels/drugs/etc.

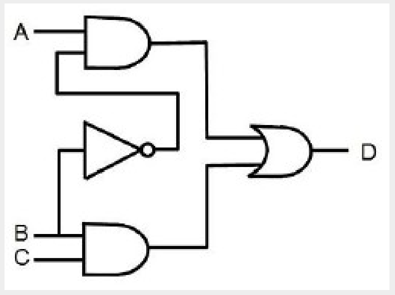
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

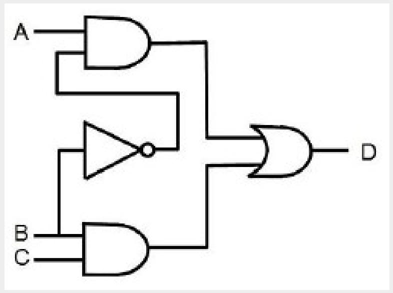
Boolean logic



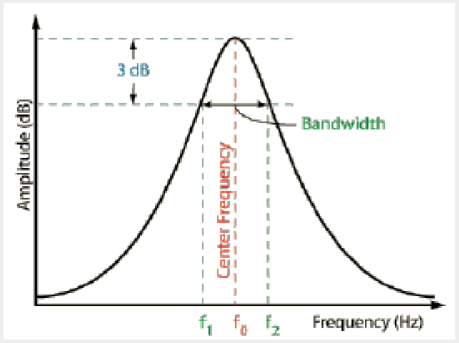
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

Boolean logic



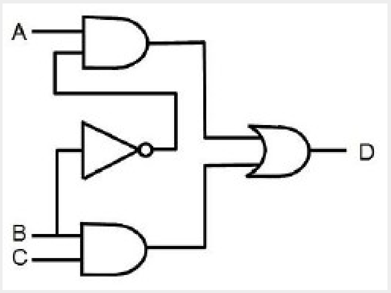
signal processing



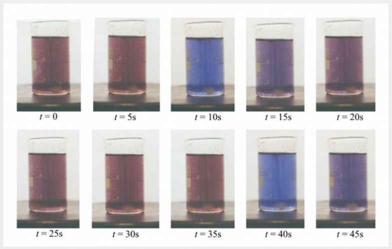
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

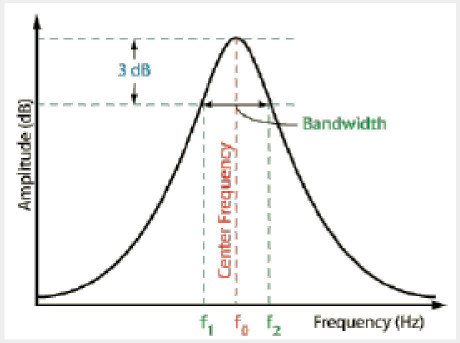
Boolean logic



oscillation



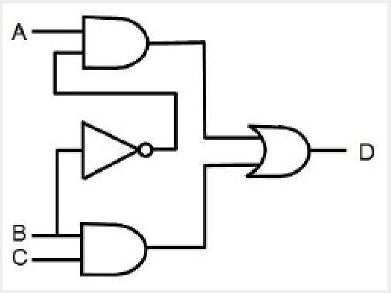
signal processing



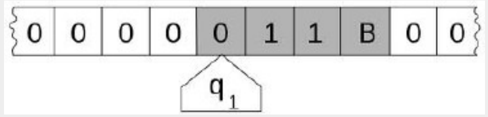
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

Boolean logic

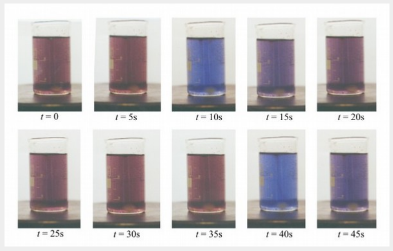


discrete algorithms

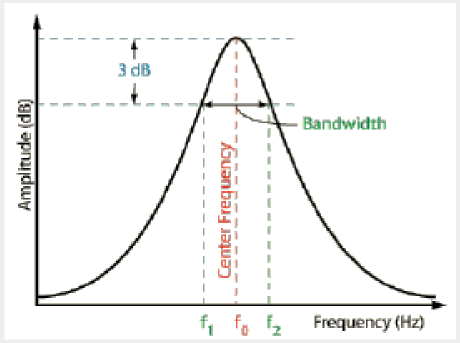


analog computing

oscillation



signal processing



Integer-valued kinetic CRN model

Integer-valued kinetic CRN model

- **species**: {*X*, *Y*, …}

Integer-valued kinetic CRN model

- **species**: $\{X, Y, \ldots\}$
- **reactions**:

$$X \xrightarrow{k_1} W + 2Y + Z$$

$$A + B \xrightarrow{k_2} X$$

Integer-valued kinetic CRN model

- **species**: {*X*, *Y*, …}
- **state**: integer vector of *counts* **s** = (#*X*, #*Y*, ...)
- **reactions**:

$$X \xrightarrow{k_1} W + 2Y + Z$$

$$A + B \xrightarrow{k_2} X$$

Integer-valued kinetic CRN model

- **species**: $\{X, Y, \ldots\}$
- **reactions**:

- **state**: integer vector of *counts* $\mathbf{s} = (\#X, \#Y, \ldots)$
- **rate** of reaction:

$X \xrightarrow{k_1} W + 2Y + Z$	$k_1 \bullet \#X$
$A + B \xrightarrow{k_2} X$	$k_2 \bullet \#A \bullet \#B$ / volume

Integer-valued kinetic CRN model

- **species**: {*X*, *Y*, …}
- **reactions**:
- **state**: integer vector of *counts* **s** = (#*X*, #*Y*, ...)
- **rate** of reaction:

$X \xrightarrow{k_1} W + 2Y + Z$	$k_1 \bullet \#X$
$A + B \xrightarrow{k_2} X$	$k_2 \bullet \#A \bullet \#B$ / volume

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

Integer-valued kinetic CRN model

- **species**: {*X*, *Y*, …}
- **state**: integer vector of *counts* **s** = (#*X*, #*Y*, ...)
- **reactions**:
- **rate** of reaction:

$X \xrightarrow{k_1} W + 2Y + Z$	$k_1 \bullet \#X$
$A + B \xrightarrow{k_2} X$	$k_2 \bullet \#A \bullet \#B$ / volume

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

E[time until next reaction] = 1 / 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:	$\{x\ X, 0\ Y\}$

CRN function computation (example)

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

input species: X

output species: Y

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

reactions: $X \underset{1}{\overset{1}{\rightleftarrows}} Y$

CRN function computation (example)

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

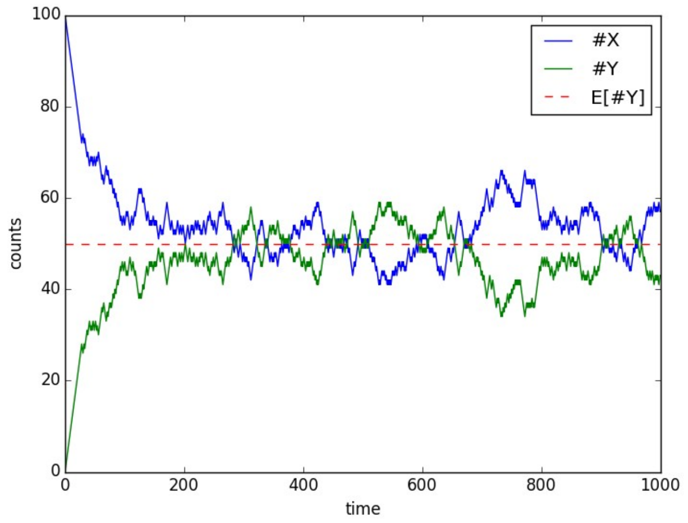
input species: X

output species: Y

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

reactions: $X \underset{1}{\overset{1}{\rightleftarrows}} Y$

#Y = *x*/2 expected at equilibrium (unstable)



CRN function computation (example)

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

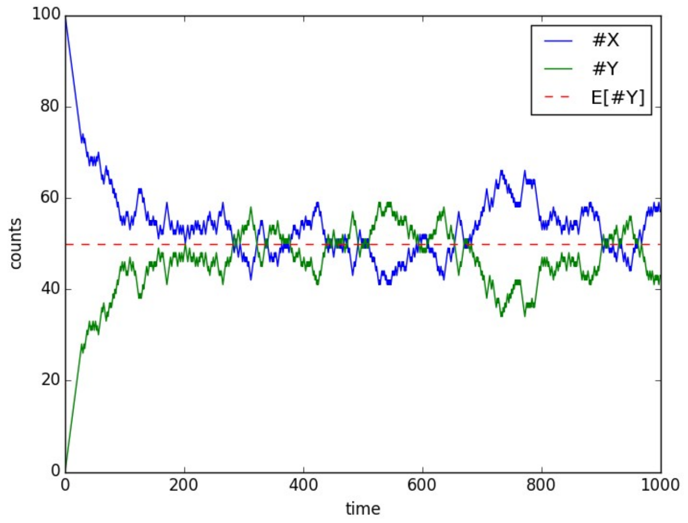
input species: X

output species: Y

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

reactions: $X \underset{1}{\overset{1}{\rightleftarrows}} Y$

#Y = *x*/2 expected at equilibrium (unstable)



reactions: $X \xrightarrow{1} Y$

$X \xrightarrow{1}$

CRN function computation (example)

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

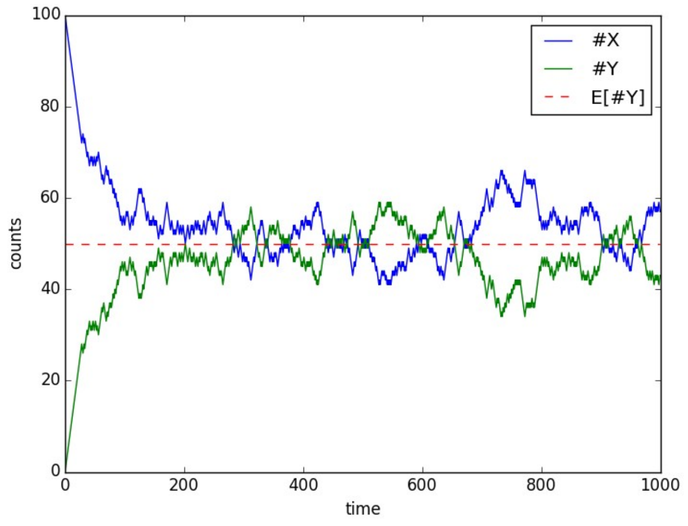
input species: X

output species: Y

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

reactions: $X \underset{1}{\overset{1}{\rightleftarrows}} Y$

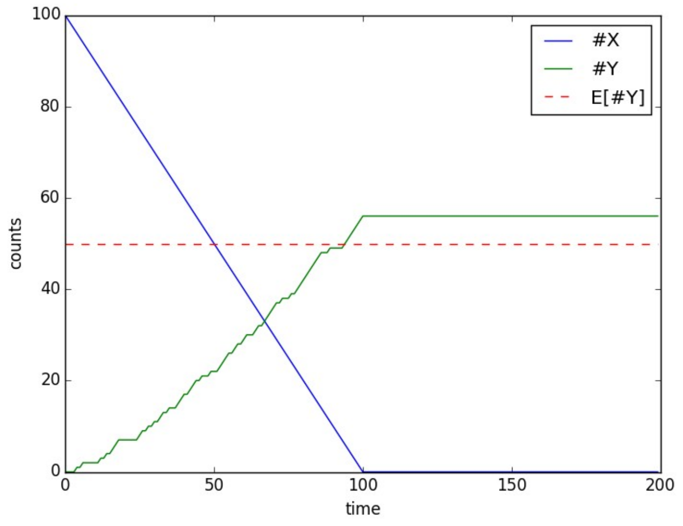
#Y = x/2 expected at equilibrium (unstable)



reactions: $X \xrightarrow{1} Y$

$X \xrightarrow{1}$

#Y stabilizes, with expected value x/2



CRN function computation (example)

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

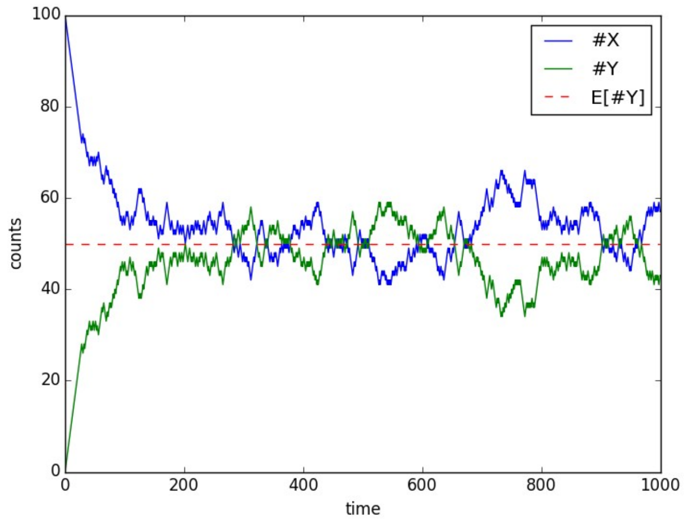
input species: X

output species: Y

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

reactions: $X \underset{1}{\overset{\cancel{1}\ 2}{\rightleftarrows}} Y$

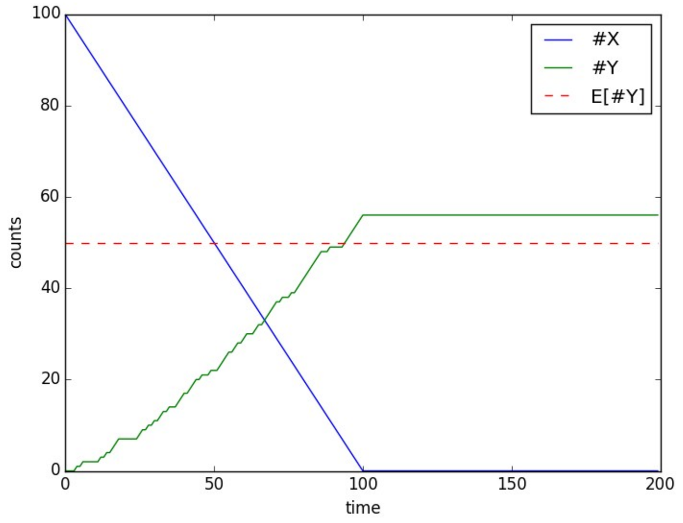
#Y = ~~$x/2$~~ $x/3$ expected at equilibrium (unstable)



reactions: $X \xrightarrow{1} Y$

$X \xrightarrow{\cancel{1}\ 2}$

#Y stabilizes, with expected value ~~$x/2$~~ $x/3$



CRN function computation (example)

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

CRN function computation (example)

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

reactions: $2X \xrightarrow{1} 2X + Y$

$Y \xrightarrow{1}$

CRN function computation (example)

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

reactions: $2X \xrightarrow{1} 2X + Y$ rate = $\#X^2$

$Y \xrightarrow{1}$ rate = $\#Y$

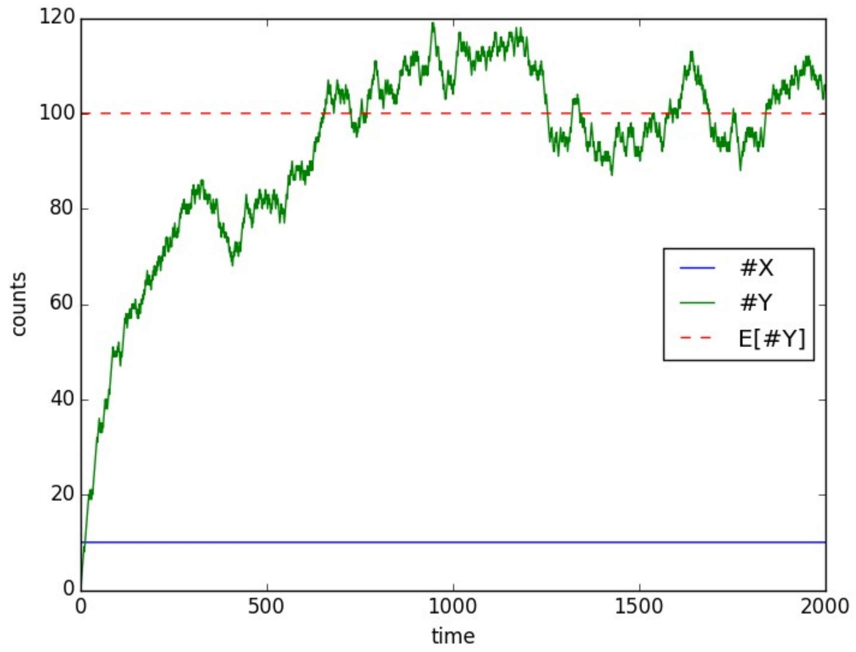
CRN function computation (example)

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

reactions: $2X \xrightarrow{1} 2X + Y$ rate = $\#X^2$

|| at equilibrium

$Y \xrightarrow{1}$ rate = $\#Y$



Rate-independent CRN computation

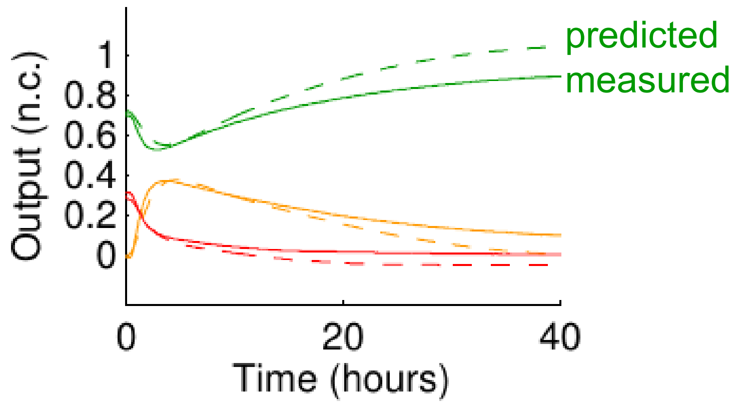
What can CRNs compute when we don't know/can't control the rates?

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!!

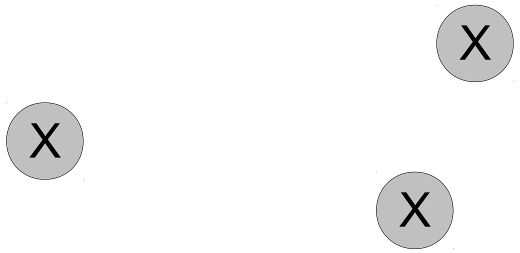
CRN function computation (example)

function: $f(x) = 2x$

CRN function computation (example)

function: $f(x) = 2x$

input species:	X
output species:	Y



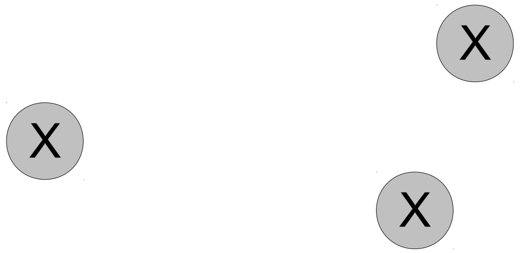
CRN function computation (example)

function: $f(x) = 2x$

input species: X

output species: Y

reactions: ??



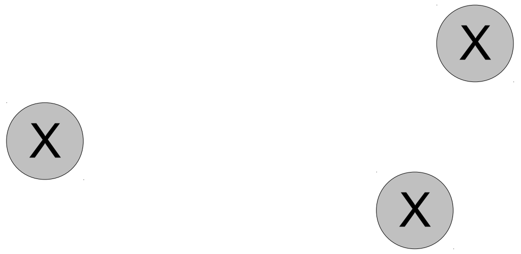
CRN function computation (example)

function: $f(x) = 2x$

input species: X

output species: Y

reactions: $X \rightarrow 2Y$

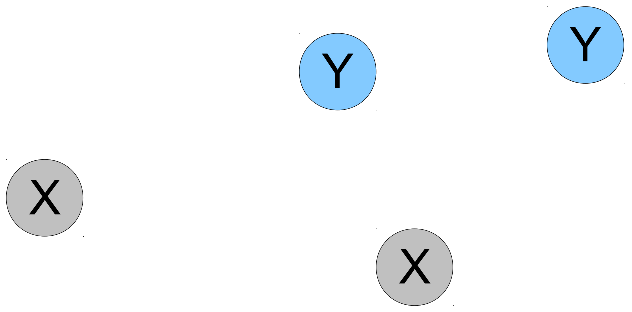


CRN function computation (example)

function: $f(x) = 2x$

input species: X
output species: Y

reactions: $X \rightarrow 2Y$

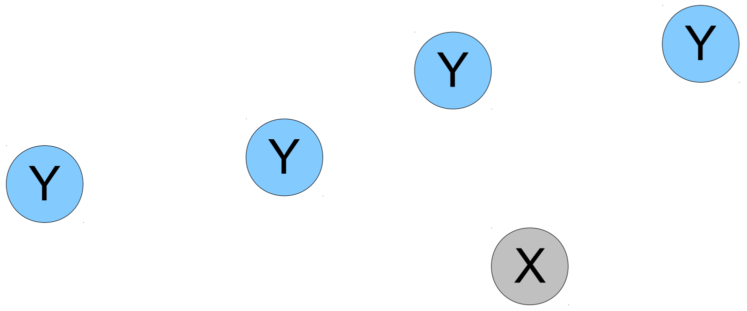


CRN function computation (example)

function: $f(x) = 2x$

input species: X
output species: Y

reactions: $X \rightarrow 2Y$



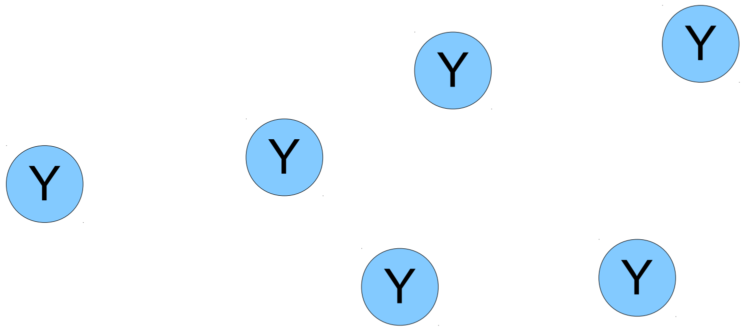
CRN function computation (example)

function: $f(x) = 2x$

input species: X

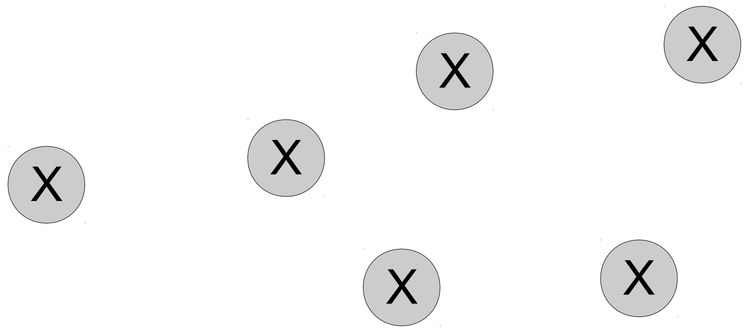
output species: Y

reactions: $X \to 2Y$



CRN function computation (example)

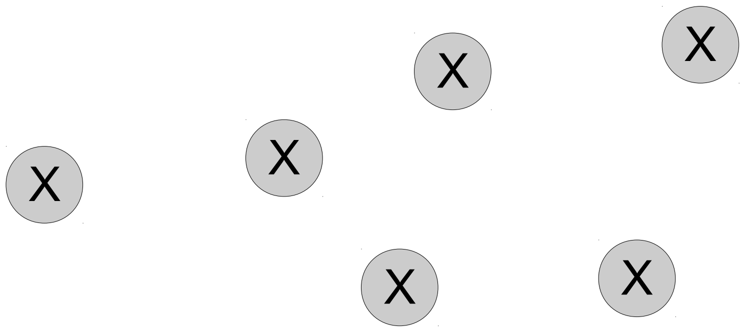
function: $f(x) = x/2$



CRN function computation (example)

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

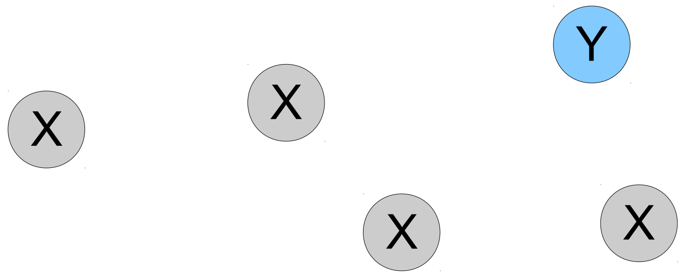
reactions: $2X \rightarrow Y$



CRN function computation (example)

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

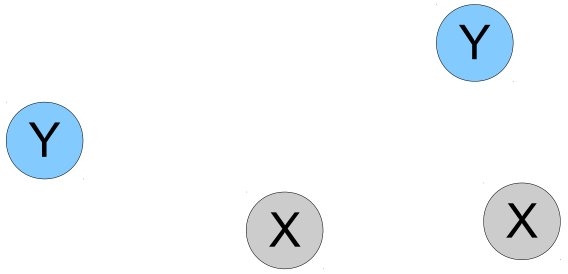
reactions: $2X \rightarrow Y$



CRN function computation (example)

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

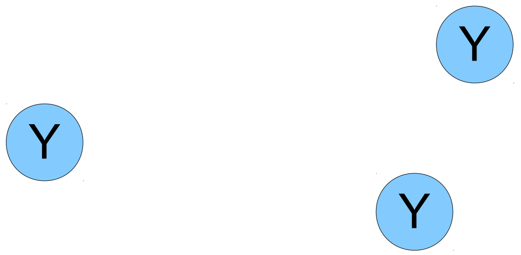
reactions: $2X \rightarrow Y$



CRN function computation (example)

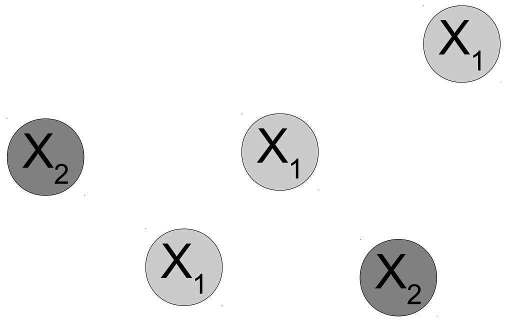
function: $f(x) = x/2$

reactions: $2X \rightarrow Y$



CRN function computation (example)

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

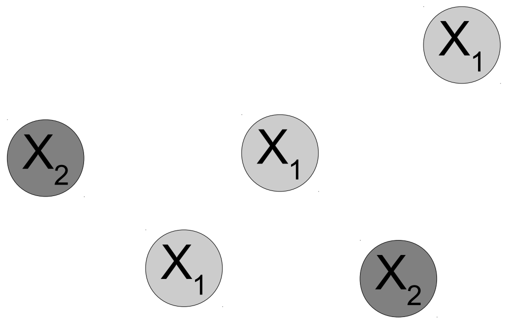


CRN function computation (example)

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

reactions: $X_1 \rightarrow Y$

$X_2 \rightarrow Y$

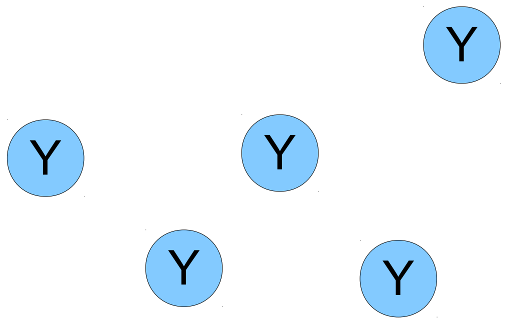


CRN function computation (example)

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

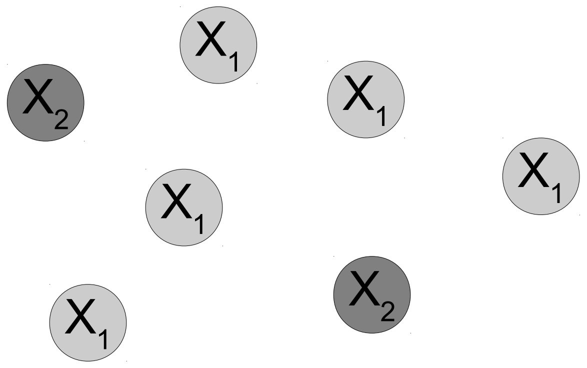
reactions: $X_1 \rightarrow Y$

$X_2 \rightarrow Y$



CRN function computation (example)

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

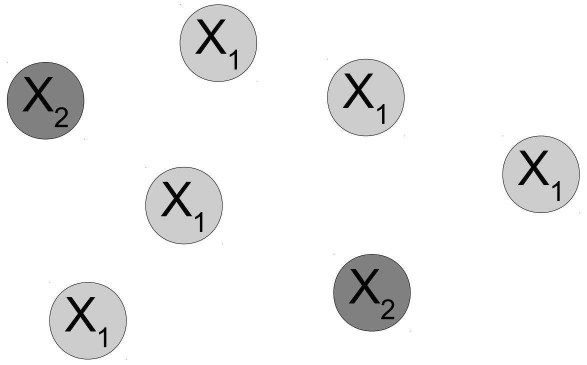


CRN function computation (example)

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

reactions: $X_1 \rightarrow Y$

$X_2 + Y \rightarrow$

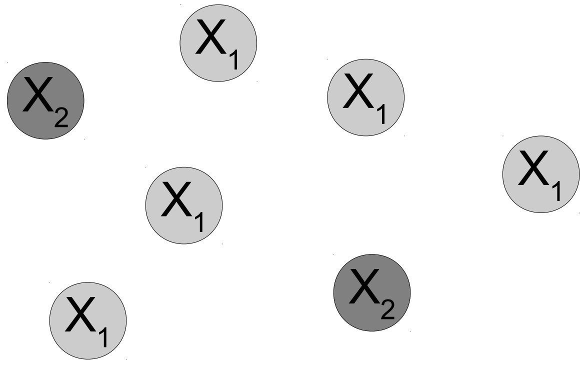


CRN function computation (example)

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

reactions: $\boxed{X_1 \rightarrow Y}$

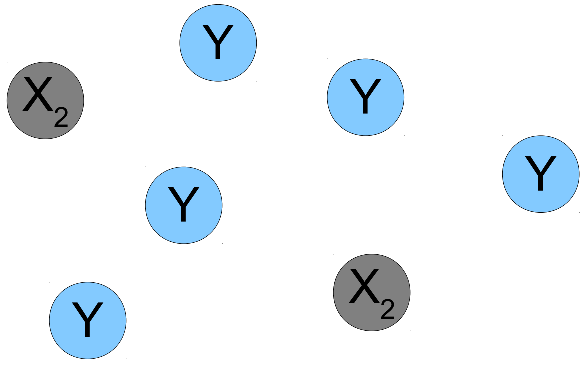
$X_2 + Y \rightarrow$



CRN function computation (example)

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

reactions: $\boxed{X_1 \rightarrow Y}$
$X_2 + Y \rightarrow$

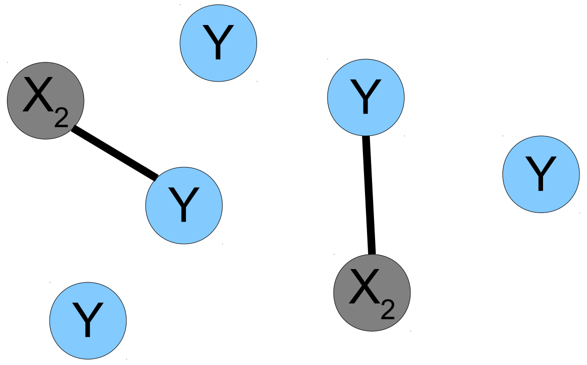


CRN function computation (example)

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

reactions: $X_1 \rightarrow Y$

$\boxed{X_2 + Y \rightarrow}$

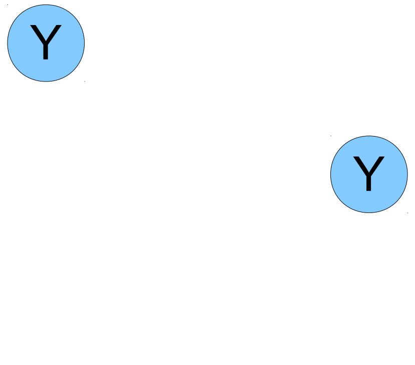


CRN function computation (example)

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

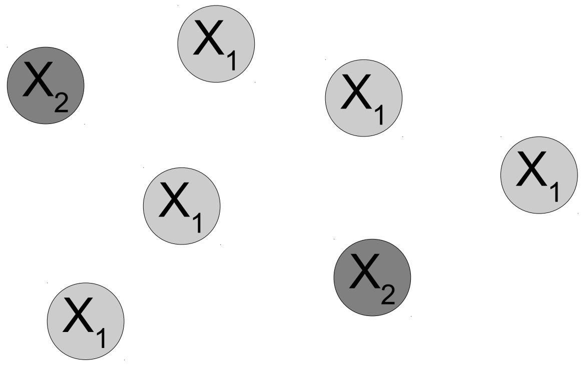
reactions: $X_1 \rightarrow Y$

$\boxed{X_2 + Y \rightarrow}$



CRN function computation (example)

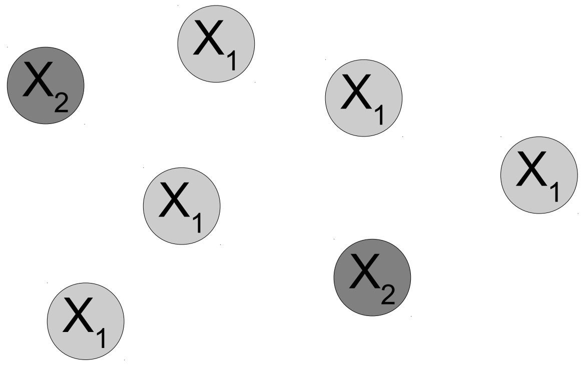
function: $f(x_1,x_2) = \min\{x_1,x_2\}$



CRN function computation (example)

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

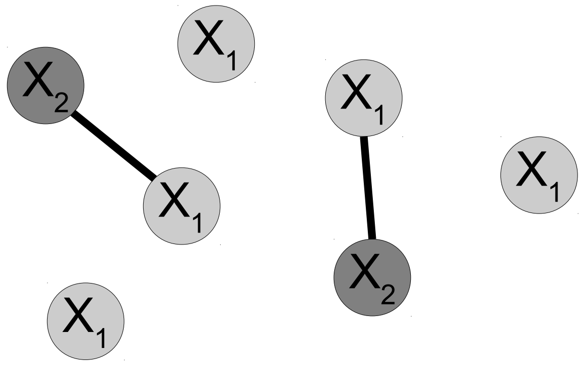
reactions: $X_1 + X_2 \rightarrow Y$



CRN function computation (example)

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

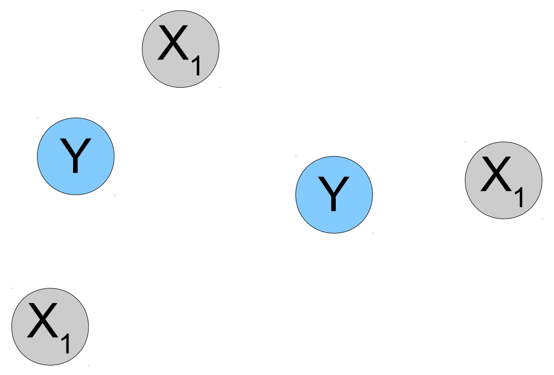
reactions: $X_1 + X_2 \rightarrow Y$



CRN function computation (example)

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

reactions: $X_1 + X_2 \rightarrow Y$

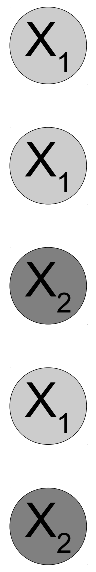


CRN function computation (example)

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

CRN function computation (example)

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

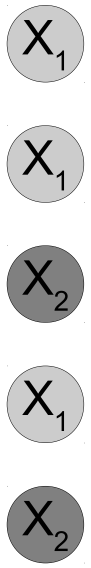


CRN function computation (example)

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

reactions:

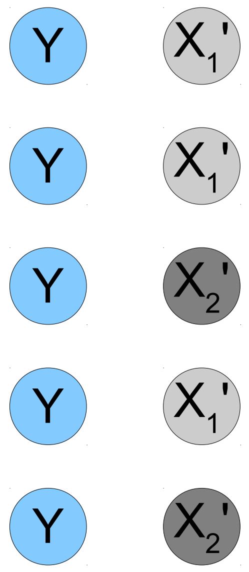
$$\boxed{\begin{array}{l} X_1 \to Y + X_1' \\ X_2 \to Y + X_2' \end{array}}$$



CRN function computation (example)

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

reactions:
$$\boxed{\begin{aligned} X_1 &\rightarrow Y + X_1' \\ X_2 &\rightarrow Y + X_2' \end{aligned}}$$

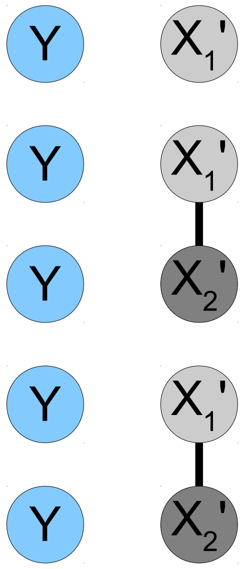


CRN function computation (example)

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

reactions:

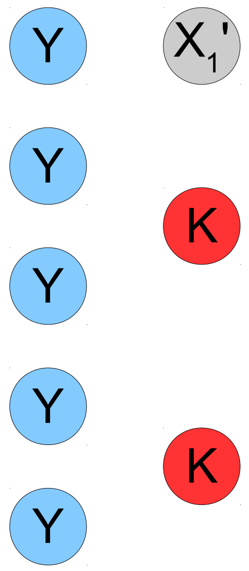
$$X_1 \rightarrow Y + X_1'$$
$$X_2 \rightarrow Y + X_2'$$
$$\boxed{X_1' + X_2' \rightarrow K}$$



CRN function computation (example)

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

reactions:
$$X_1 \rightarrow Y + X_1'$$
$$X_2 \rightarrow Y + X_2'$$
$$\boxed{X_1' + X_2' \rightarrow K}$$

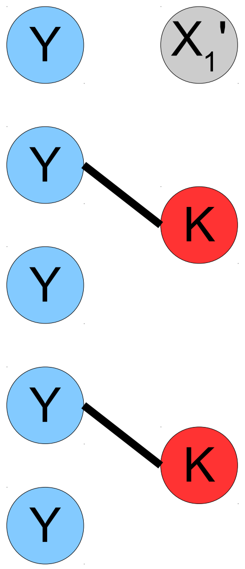


CRN function computation (example)

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

reactions:

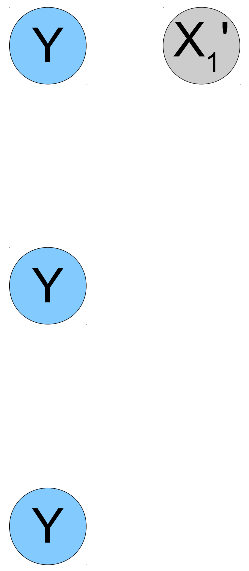
$$X_1 \to Y + X_1'$$
$$X_2 \to Y + X_2'$$
$$X_1' + X_2' \to K$$
$$\boxed{K + Y \to \quad\quad}$$



CRN function computation (example)

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

reactions:
$$X_1 \to Y + X_1'$$
$$X_2 \to Y + X_2'$$
$$X_1' + X_2' \to K$$
$$\boxed{K + Y \to}$$



Other functions?

$f(x) = x^2$?

$f(x_1, x_2) = x_1 \cdot x_2$?

$f(x) = 2^x$?

$f(x) = x \cdot \sqrt{2}$?

Other functions?

~~$f(x) = x^2$?~~

~~$f(x_1, x_2) = x_1 \cdot x_2$?~~

~~$f(x) = 2^x$?~~

~~$f(x) = x \cdot \sqrt{2}$?~~

Stable function computation (definition)

task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

Stable function computation (definition)

task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

- **initial state i**: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

Stable function computation (definition)

task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

- **initial state i**: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species
- **output of state s**: $\mathbf{s}(Y)$

Stable function computation (definition)

task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

- **initial state i**: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species
- **output of state s**: $\mathbf{s}(Y)$
- **output-stable state**: all states reachable from it have same output

Stable function computation (definition)

task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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

Stable function computation (definition)

task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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

i

Stable function computation (definition)

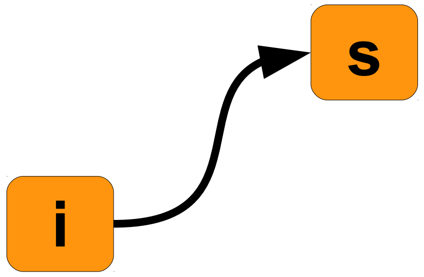
task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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



Stable function computation (definition)

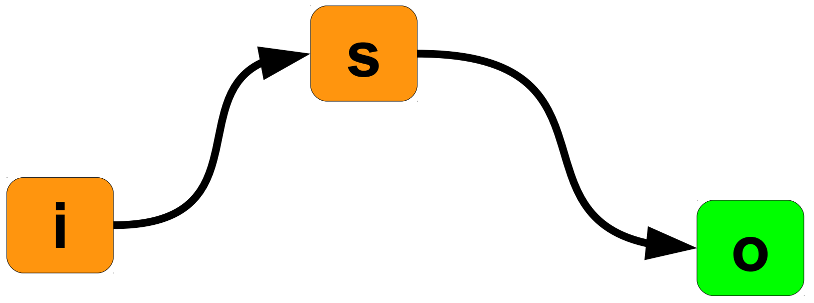
task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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



$\boldsymbol{o}(Y) = f(x_1,\ldots,x_k)$

Stable function computation (definition)

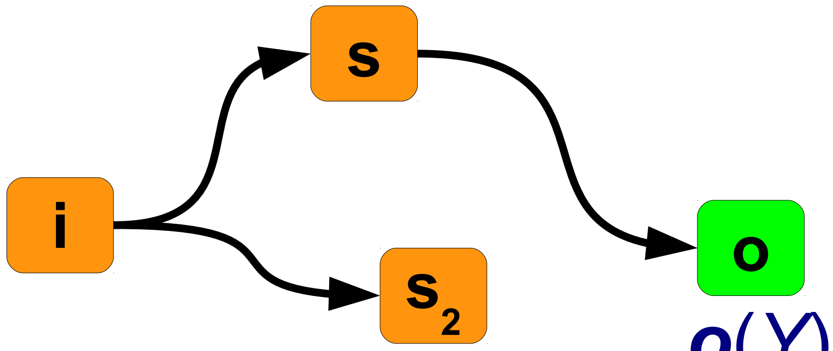
task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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



$\mathbf{o}(Y) = f(x_1,\ldots,x_k)$

Stable function computation (definition)

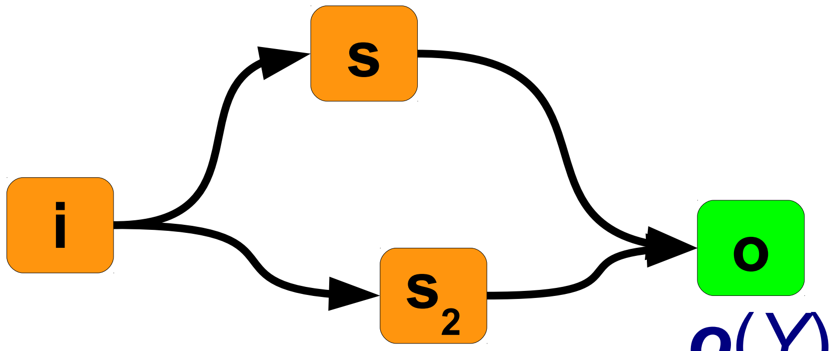
task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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


$$\begin{tikzcd} & s \arrow[dr] & \\ i \arrow[ur] \arrow[r] & s_2 \arrow[r] & o \end{tikzcd}$$

$\mathbf{o}(Y) = f(x_1,\ldots,x_k)$

Stable function computation (definition)

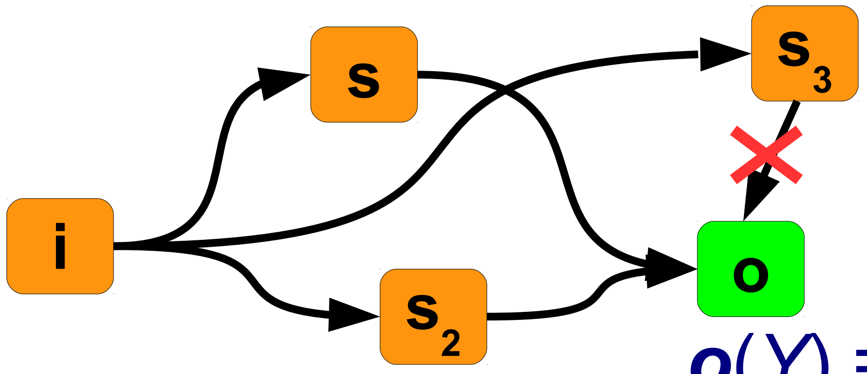
task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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



$\mathbf{o}(Y) = f(x_1,\ldots,x_k)$

Stable function computation (definition)

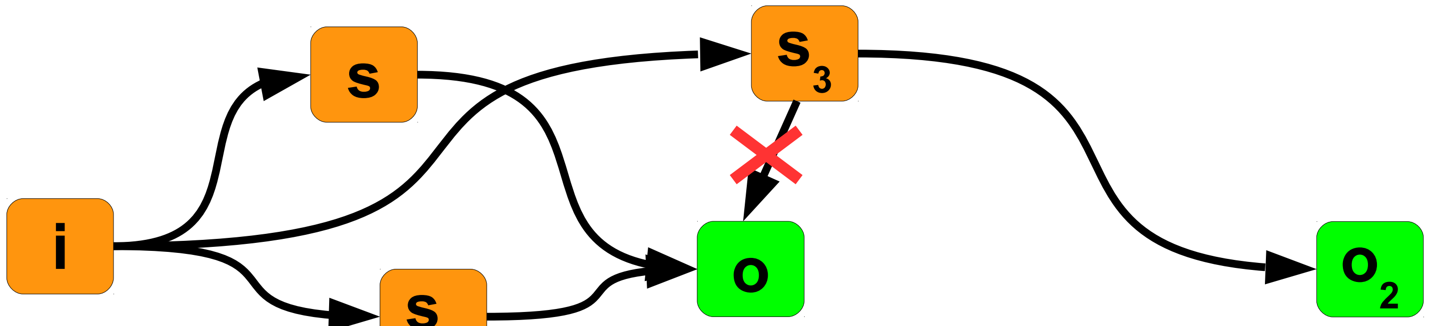
task: given $x_1,\ldots,x_k \in \mathbb{N}$, compute $f(x_1,\ldots,x_k) \in \mathbb{N}$

initial state i: $\mathbf{i}(X_1) = x_1,\ldots,\mathbf{i}(X_k) = x_k$, constant counts of other species

output of state s: $\mathbf{s}(Y)$

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

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



$\mathbf{o}(Y) = f(x_1,\ldots,x_k) = \mathbf{o}_2(Y)$

Stable computation characterization

Theorem: A function is stably computed by a CRN if and only if it is *semilinear*. (≈ piecewise linear)

[Angluin, Aspnes, Diamadi, Fisher, Peralta, Principles of Distributed Computing 2004]

[Angluin, Aspnes, Eisenstat, Principles of Distributed Computing 2006]

[Chen, D, Soloveichik, DNA Computing 2012]

Real-valued CRNs

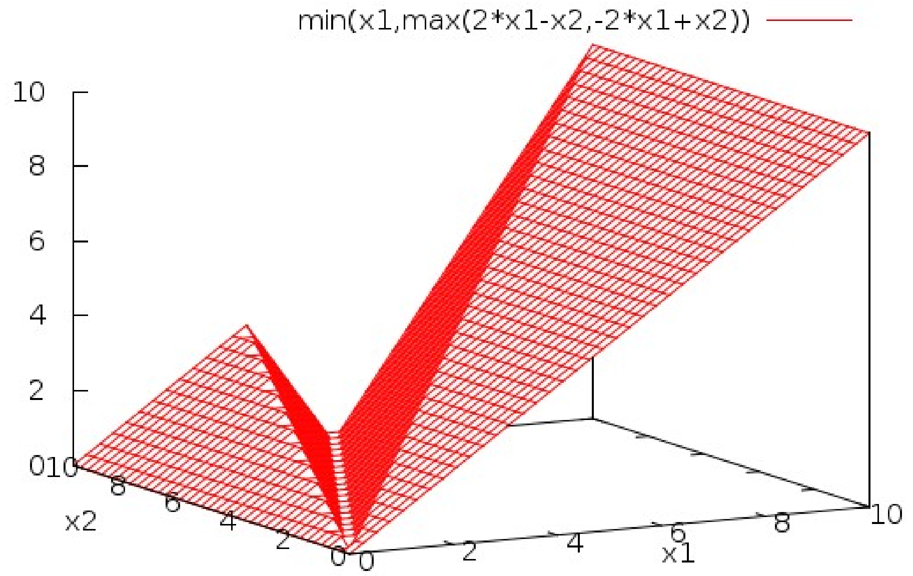
Theorem from previous slide: A function is stably computed by a integer-valued CRN if and only if it is *semilinear*.

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*.

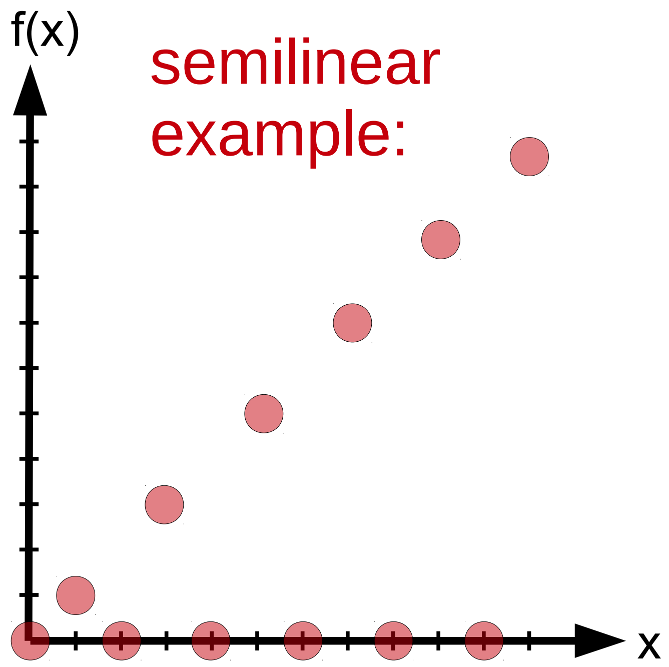
[Chen, D, Soloveichik, Innovations in Theoretical Computer Science 2014]



Real-valued CRNs

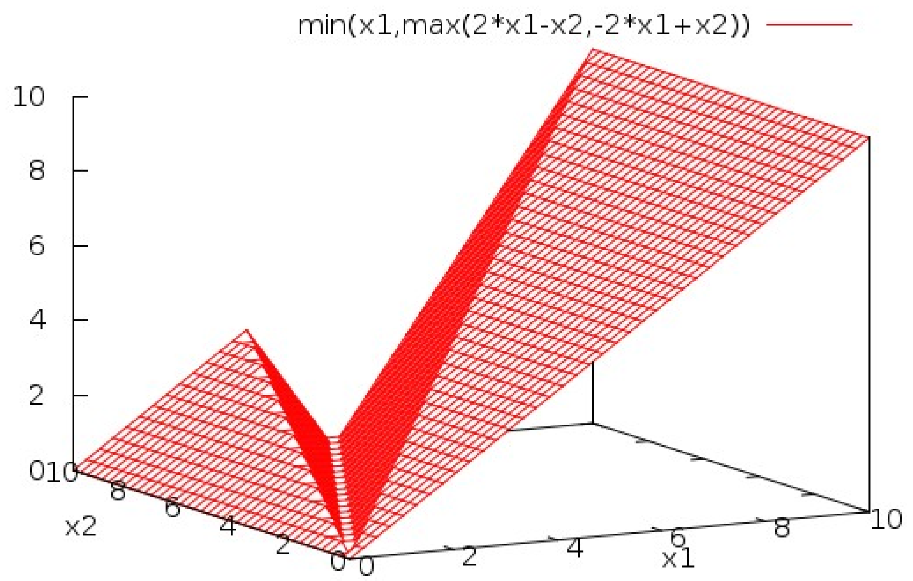
Theorem from previous slide: A function is stably computed by a integer-valued CRN if and only if it is *semilinear*.

semilinear ≈ 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 *continuous* and *piecewise linear*.

[Chen, D, Soloveichik, Innovations in Theoretical Computer Science 2014]



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

[Angluin, Aspnes, Eisenstat, Symposium on Distributed Computing 2006]
[Soloveichik, Cook, Winfree, Bruck, Natural Computing 2008]

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

CRNs with small probability of error are Turing universal

[Angluin, Aspnes, Eisenstat, Symposium on Distributed Computing 2006]
[Soloveichik, Cook, Winfree, Bruck, Natural Computing 2008]

(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.*

CRNs with small probability of error are Turing universal

[Angluin, Aspnes, Eisenstat, Symposium on Distributed Computing 2006]
[Soloveichik, Cook, Winfree, Bruck, Natural Computing 2008]

(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.*

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

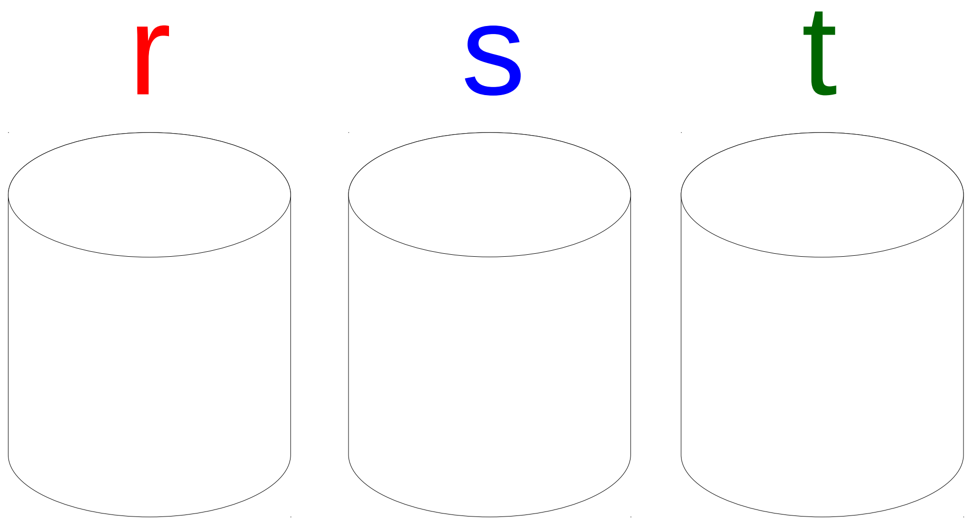
Implication: *General CRN long-term behavior cannot be predicted faster than by simulating.*

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

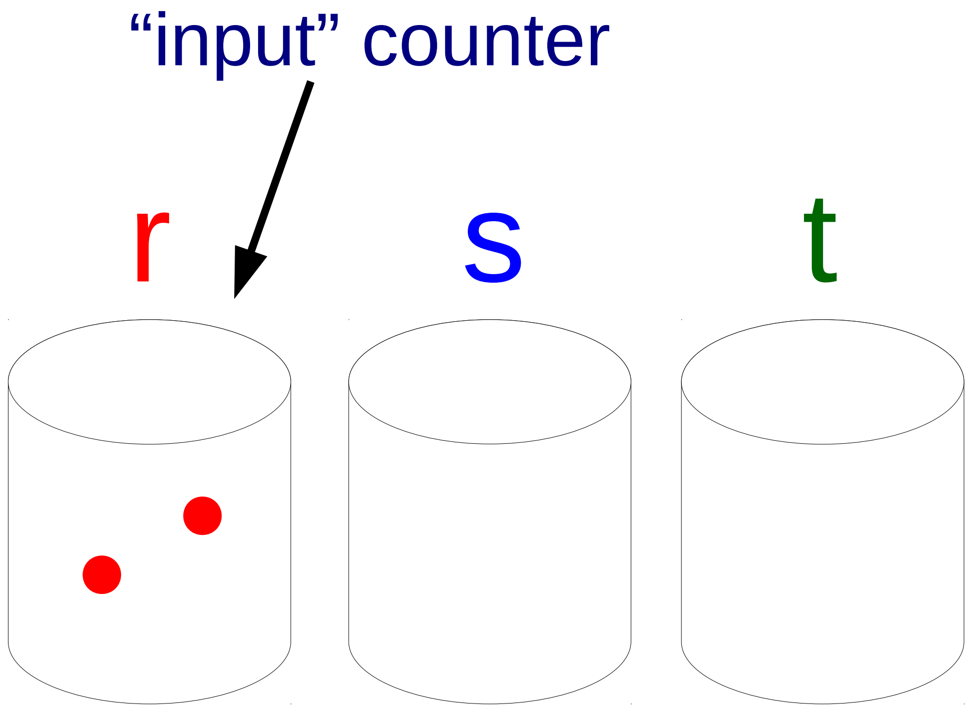
Implication: *General CRN long-term behavior cannot be*

Counter (register) machine

Counter (register) machine

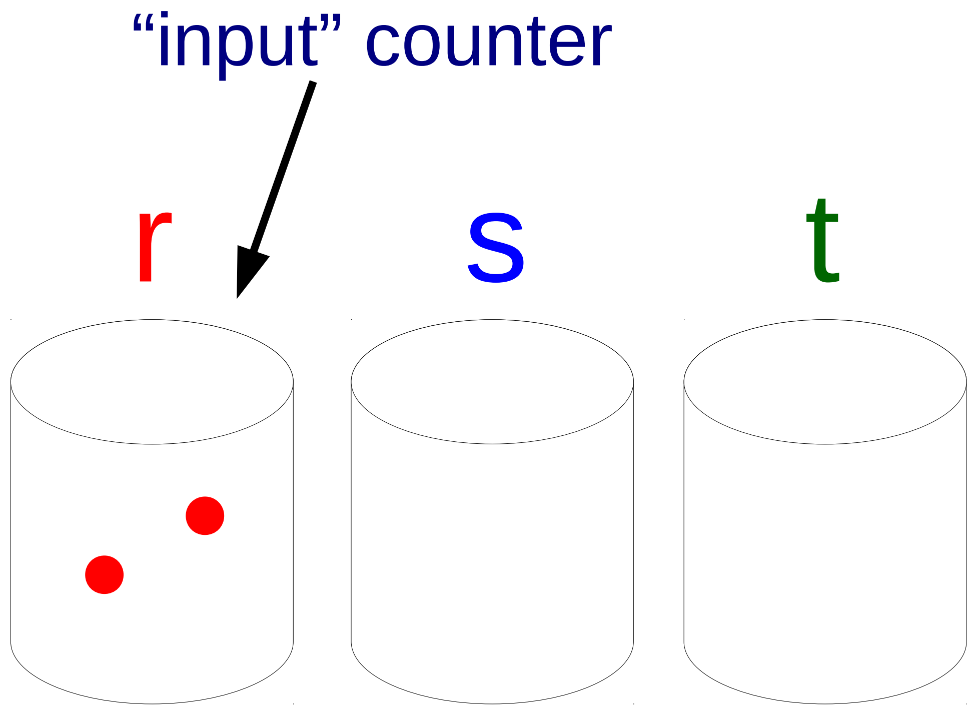


Counter (register) machine



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) *dec*(r)

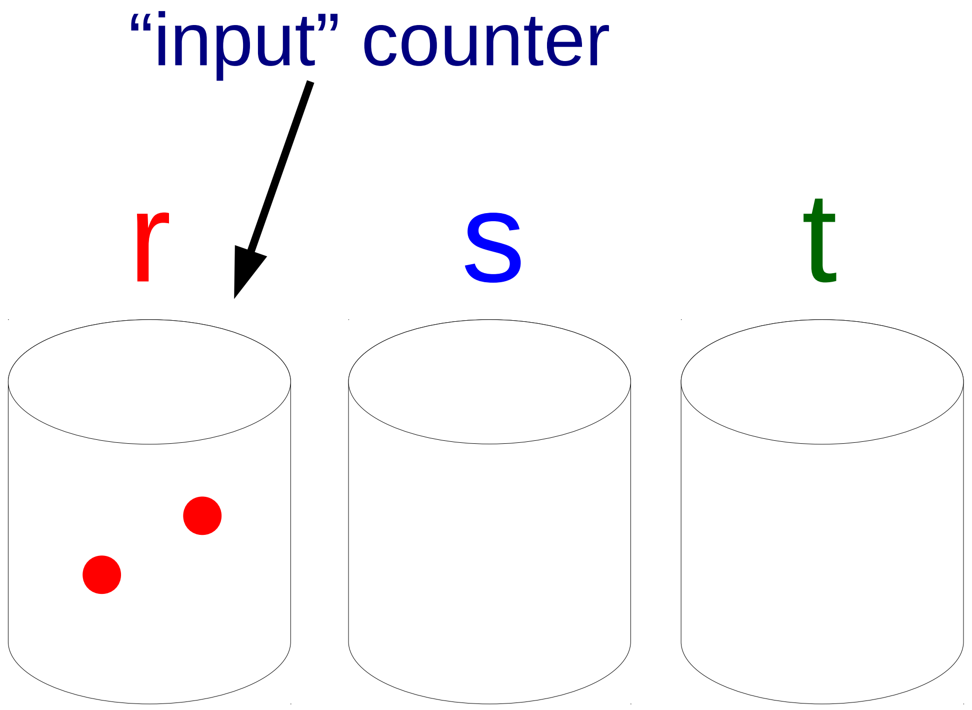
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

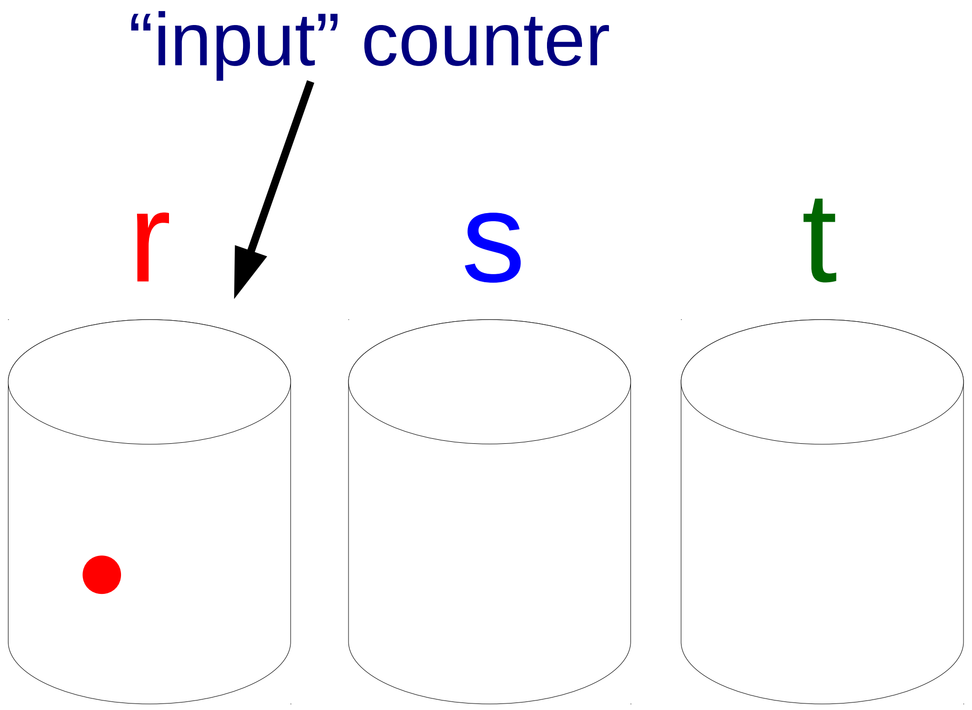
5) *dec*(t)

6) *inc*(s)



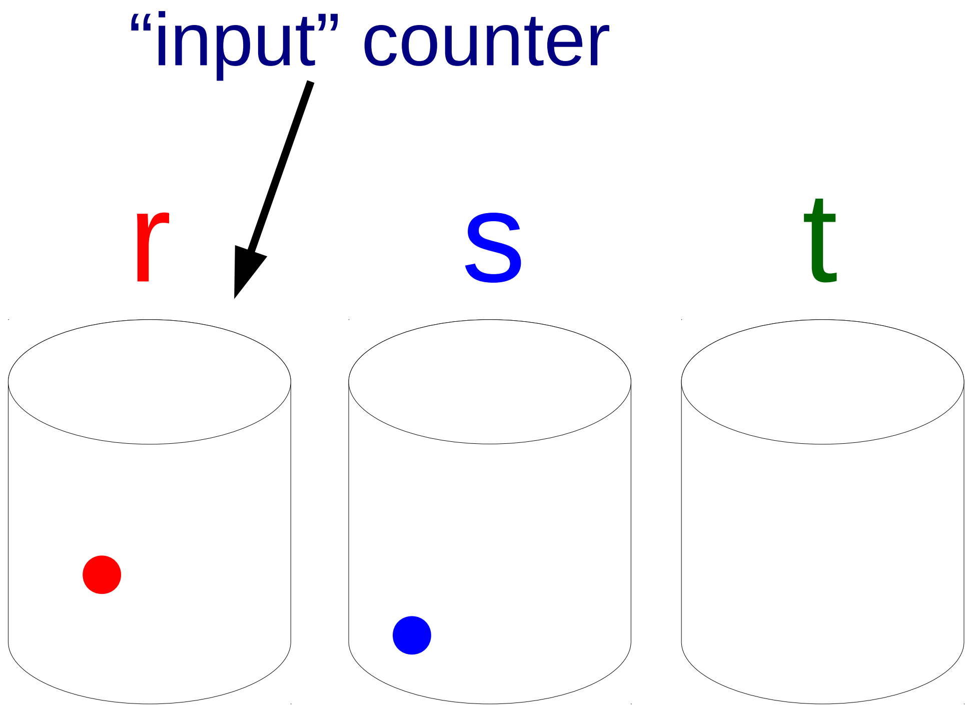
Counter (register) machine

1) *dec*(r)
2) *inc*(s)
3) *inc*(s)
4) *inc*(s)
5) *dec*(t)
6) *inc*(s)



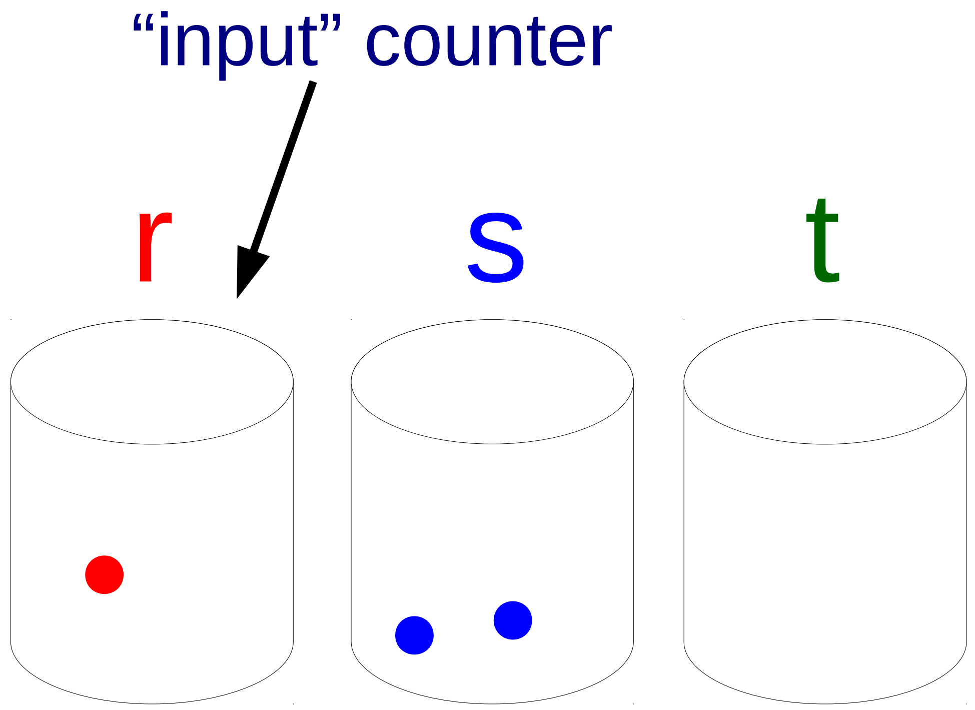
Counter (register) machine

1) *dec*(r)
2) *inc*(s)
3) *inc*(s)
4) *inc*(s)
5) *dec*(t)
6) *inc*(s)



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) *dec*(r)

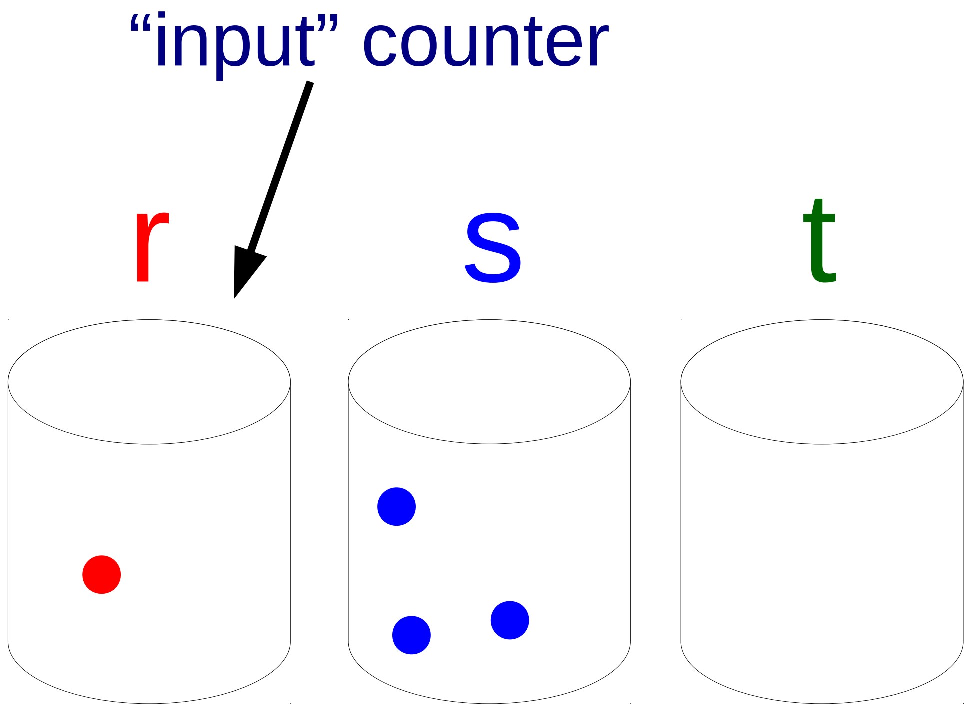
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

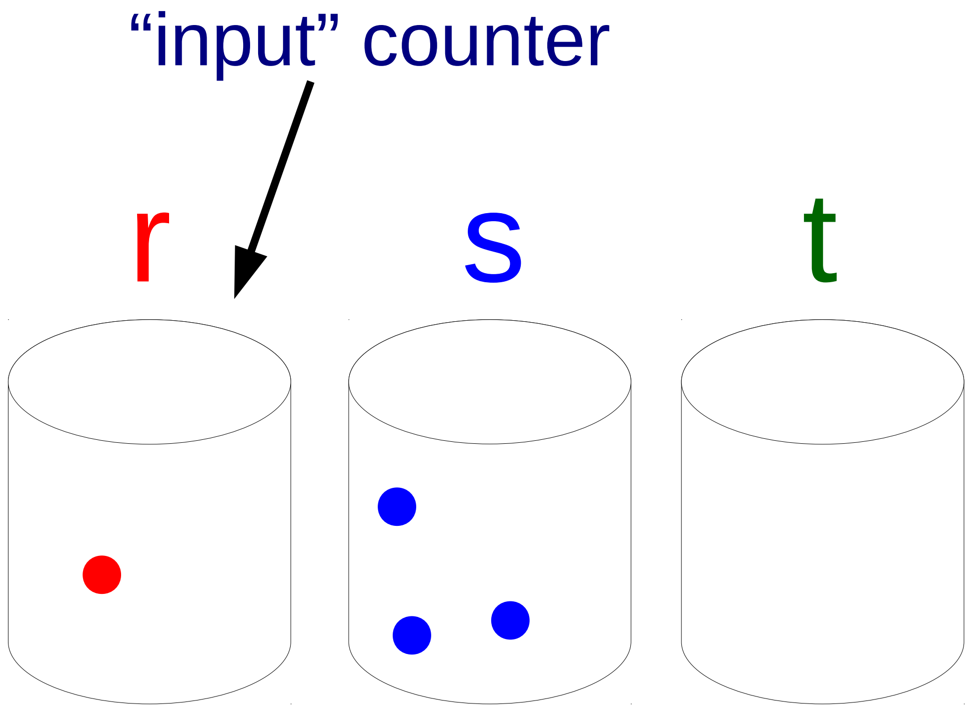
5) *dec*(t)

6) *inc*(s)



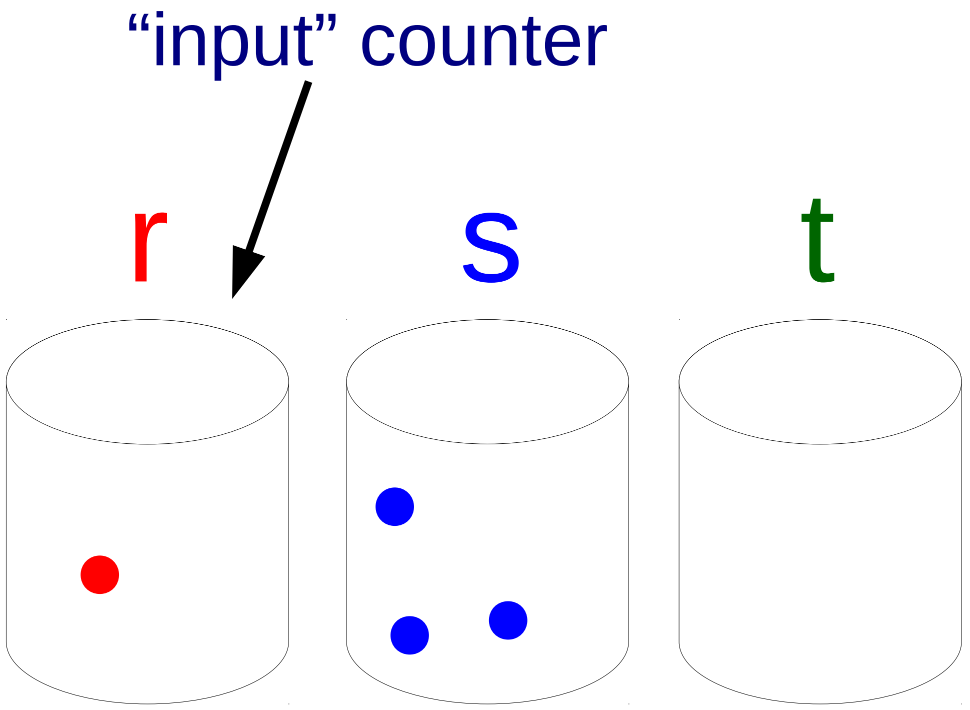
Counter (register) machine

1) *dec*(r)
2) *inc*(s)
3) *inc*(s)
4) *inc*(s)
5) *dec*(t)
6) *inc*(s)



Counter (register) machine

1) *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) *dec*(r) if empty goto 6

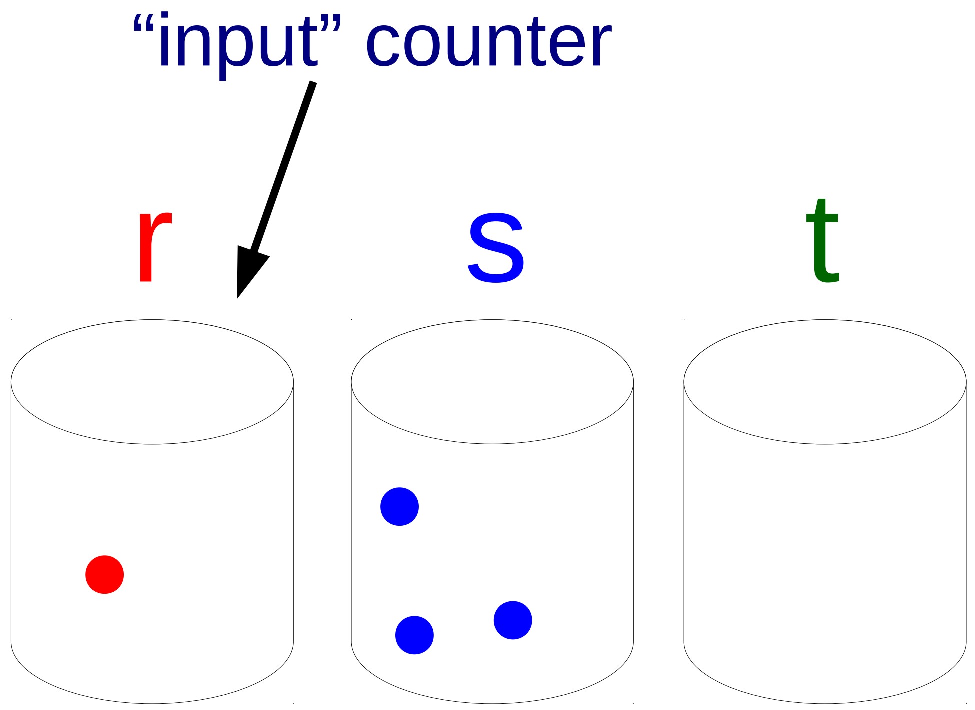
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

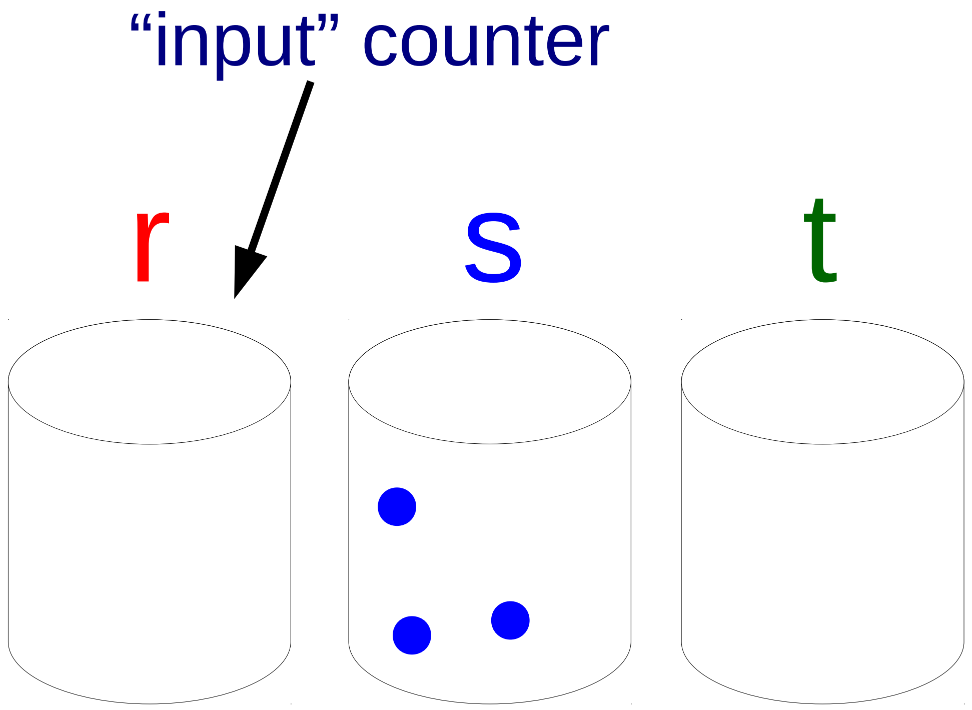
5) *dec*(t) if empty goto 1

6) *inc*(s)



Counter (register) machine

1) *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) *dec*(r) if empty goto 6

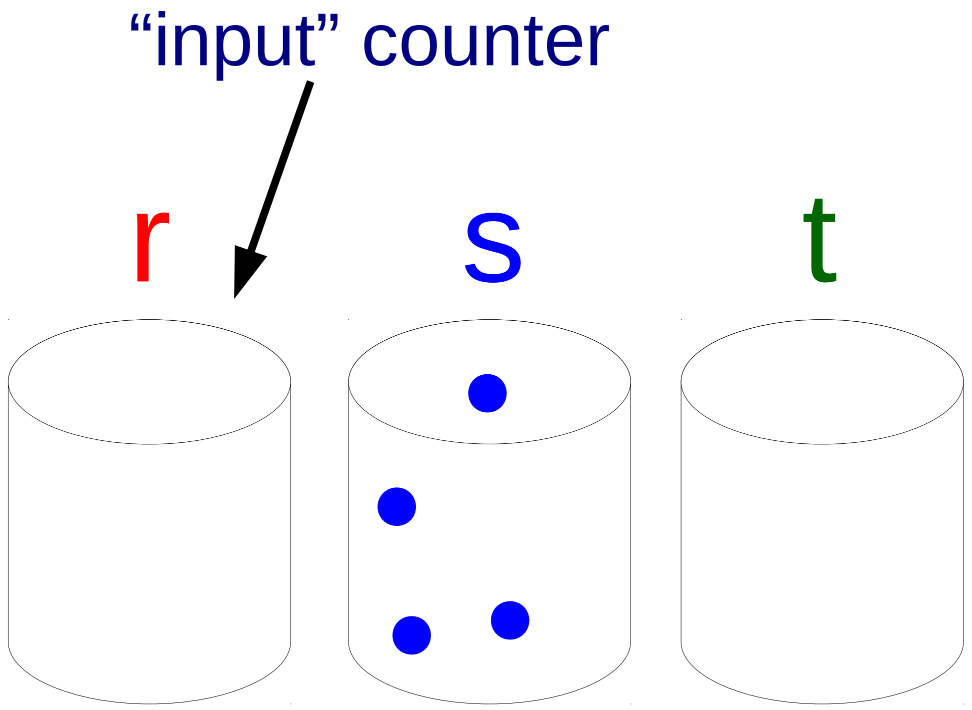
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

5) *dec*(t) if empty goto 1

6) *inc*(s)



Counter (register) machine

1) *dec*(r) if empty goto 6

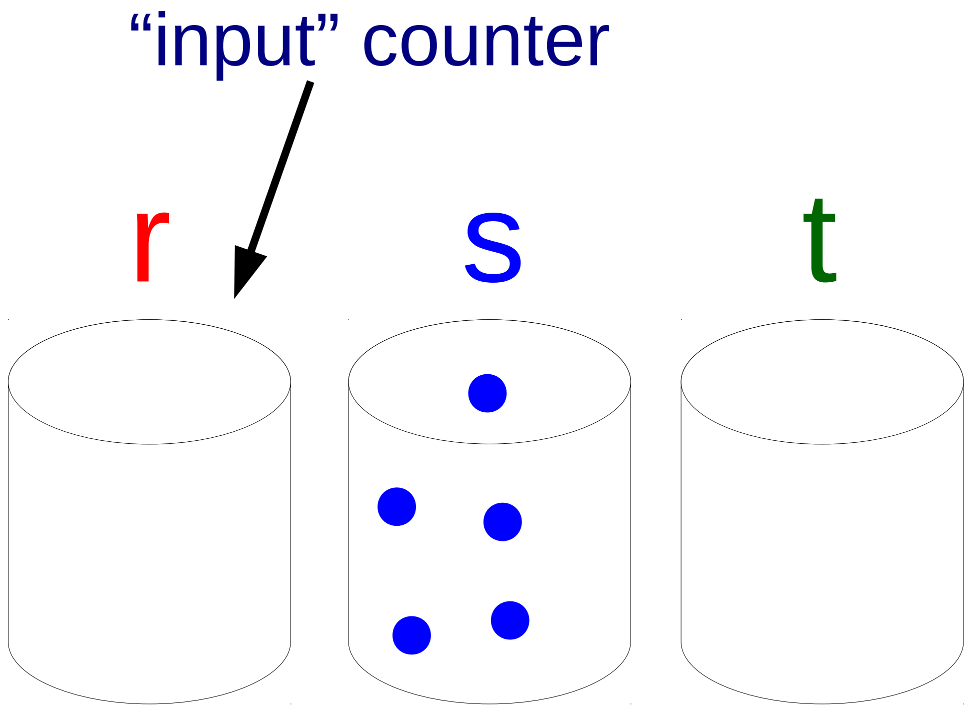
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

5) *dec*(t) if empty goto 1

6) *inc*(s)



Counter (register) machine

1) *dec*(r) if empty goto 6

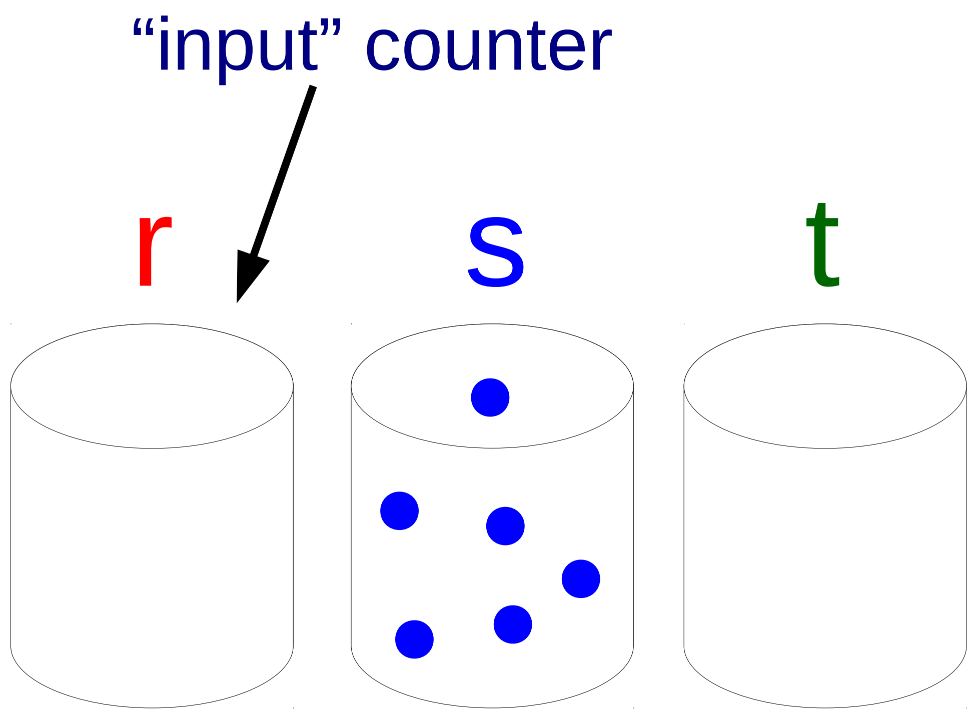
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

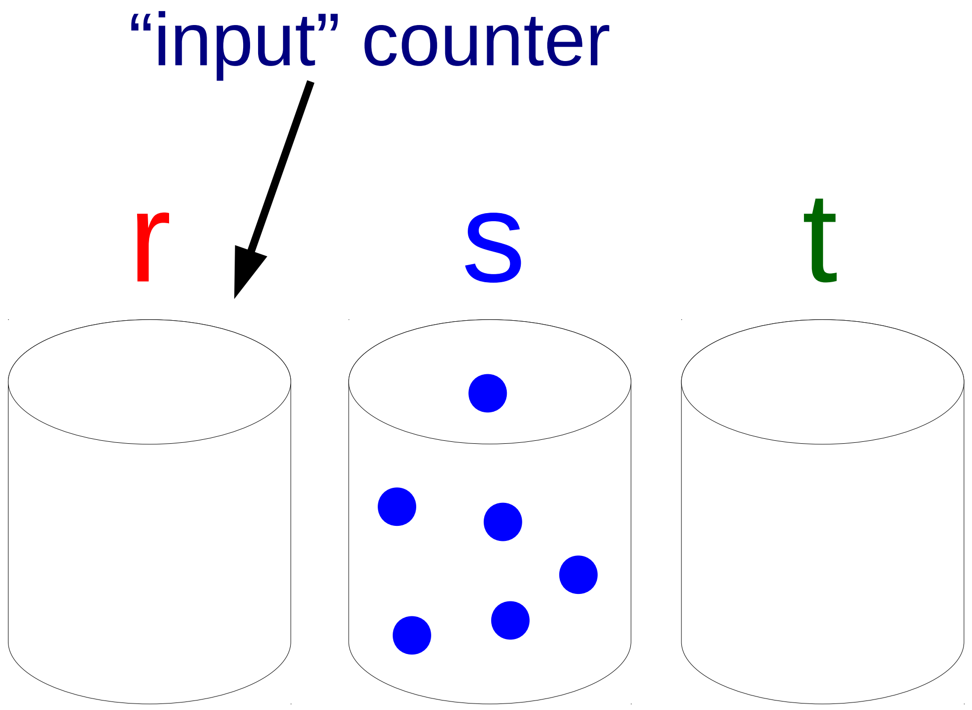
5) *dec*(t) if empty goto 1

6) *inc*(s)



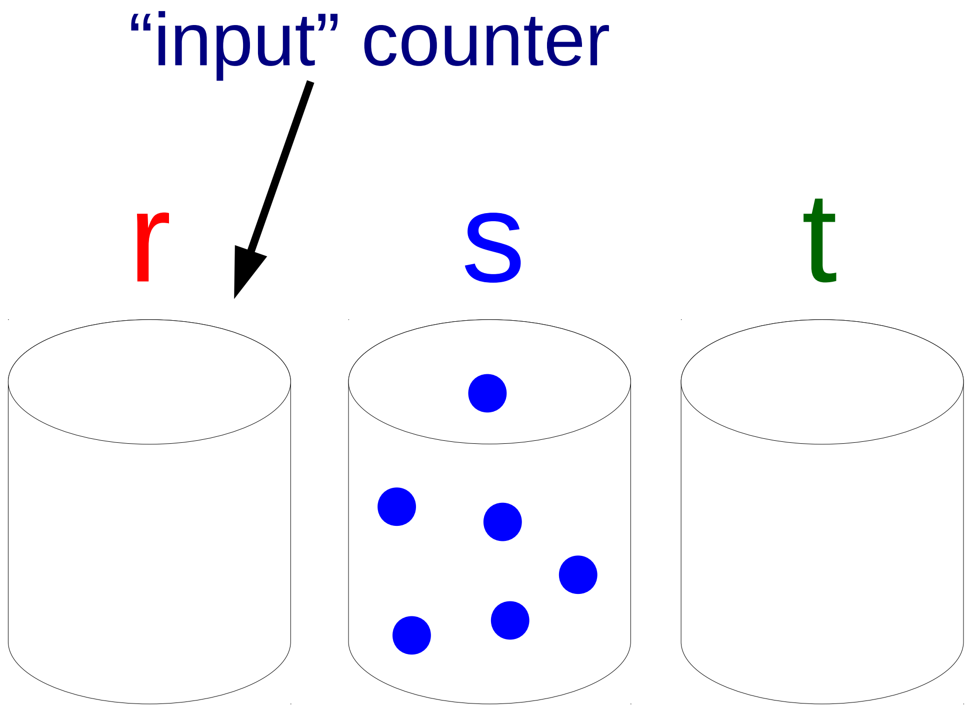
Counter (register) machine

1) *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) *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) *dec*(r) if empty goto 6

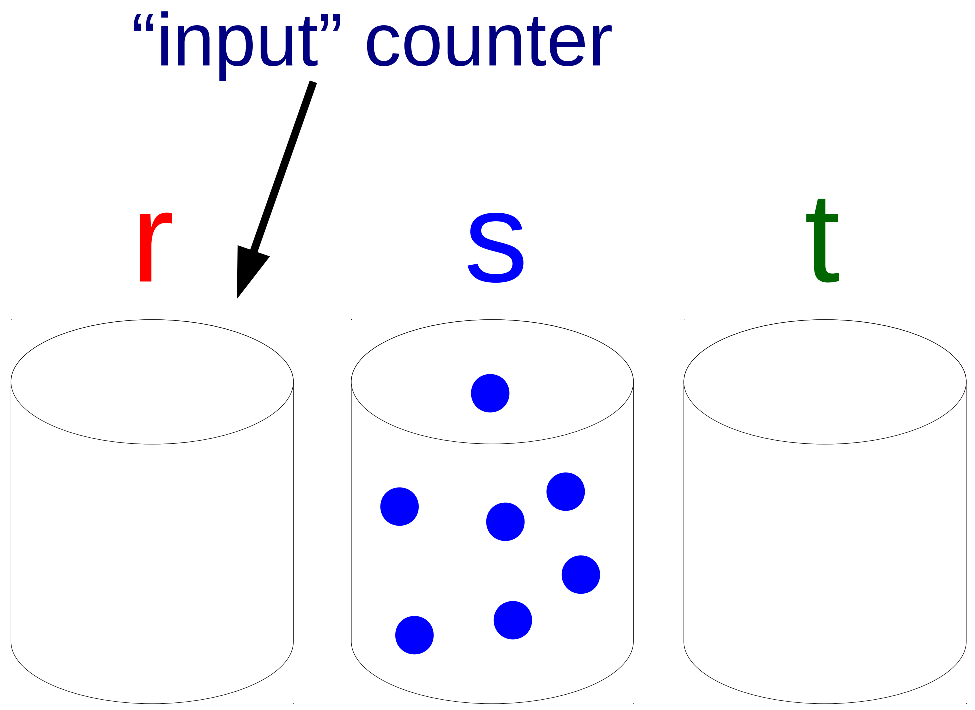
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

5) *dec*(t) if empty goto 1

6) *inc*(s)



Counter (register) machine

1) *dec*(r) if empty goto 6

2) *inc*(s)

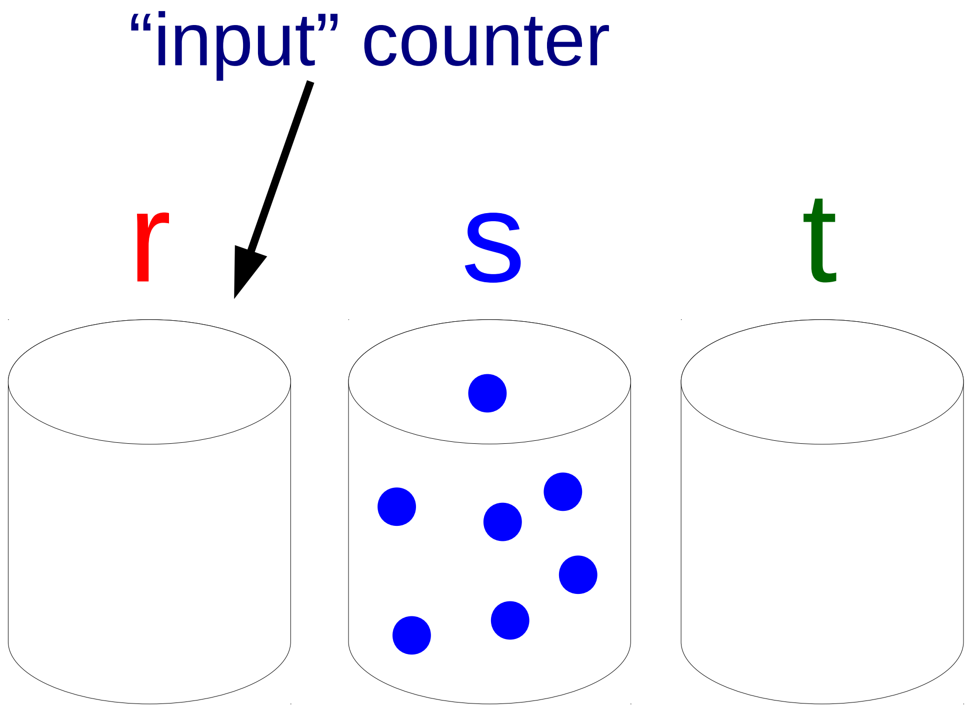
3) *inc*(s)

4) *inc*(s)

5) *dec*(t) if empty goto 1

6) *inc*(s)

HALT



Counter (register) machine

1) *dec*(r) if empty goto 6

2) *inc*(s)

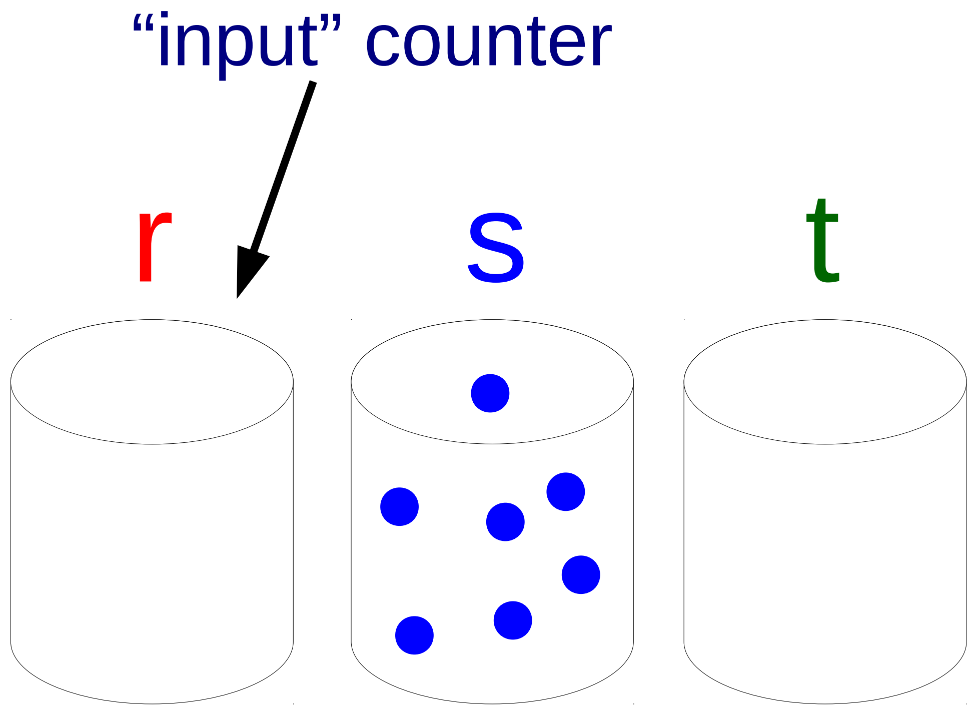
3) *inc*(s)

4) *inc*(s)

5) *dec*(t) if empty goto 1

6) *inc*(s)

HALT



computes $f(n) = 3n+1$

CRNs can simulate counter machines

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
```

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 $\{n\ R, 1\ L_1\}$

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 $\{n\ R, 1\ L_1\}$

$$L_1 \rightarrow L_2 + R$$

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 = 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 $\{n\,R, 1\,L_1\}$

$L_1 \rightarrow L_2 + R$

$L_2 + R \rightarrow L_3$; $L_2 \rightarrow L_1$

CRNs can simulate counter machines

Counter machine:	**CRN:**
r = input *n*, start line 1	initial state $\{n\ R, 1\ L_1\}$
1) *inc*(r)	$L_1 \rightarrow L_2 + R$
2) *dec*(r) if zero goto 1	$L_2 + R \rightarrow L_3 \ ; \ L_2 \rightarrow L_1$
3) *inc*(s)	$L_3 \rightarrow L_4 + S$
4) *dec*(s) if zero goto 2	$L_4 + S \rightarrow L_5 \ ; \ L_4 \rightarrow L_2$

CRNs can simulate counter machines with probability < 1

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 $\{n\,R, 1\,L_1\}$

$L_1 \rightarrow L_2 + R$

$L_2 + R \rightarrow L_3$; $L_2 \rightarrow L_1$

Need to be **very** slow!

$L_3 \rightarrow L_4 + S$

$L_4 + S \rightarrow L_5$; $L_4 \rightarrow L_2$

How to slow down reaction $L_2 \rightarrow L_1$?

How to slow down reaction $L_2 \to L_1$?

Use a clock:

1 C_0, 1 F

How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

1 C_0, 1 F

$$F + C_0 \rightarrow F + C_1$$

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

How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

$1\ C_0$, $1\ F$, $n\ B$

$$F + C_0 \rightarrow F + C_1 \qquad B + C_1 \rightarrow B + C_0$$

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

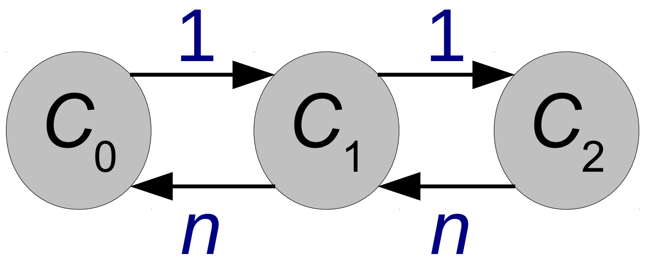
How to slow down reaction $L_2 \to L_1$?

Use a clock:

$1\ C_0$, $1\ F$, $n\ B$

$F + C_0 \to F + C_1$ $B + C_1 \to B + C_0$

$F + C_1 \to F + C_2$ $B + C_2 \to B + C_1$



$$\begin{tikzcd} C_0 \arrow[r, bend left, "1"] & C_1 \arrow[l, bend left, "n"] \arrow[r, bend left, "1"] & C_2 \arrow[l, bend left, "n"] \end{tikzcd}$$

reverse-biased random walk

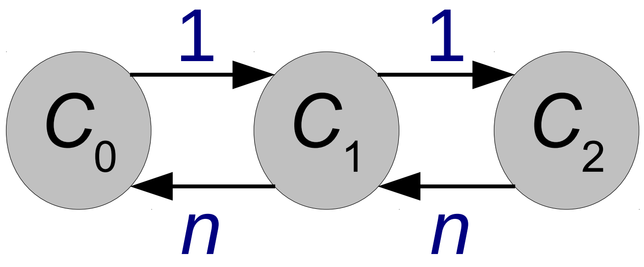
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

$1\ C_0$, $1\ F$, $n\ B$

$F + C_0 \rightarrow F + C_1$ $B + C_1 \rightarrow B + C_0$

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



$$\begin{tikzcd} C_0 \arrow[r, bend left, "1"] & C_1 \arrow[l, bend left, "n"] \arrow[r, bend left, "1"] & C_2 \arrow[l, bend left, "n"] \end{tikzcd}$$

reverse-biased random walk

C_2 appears after expected time $\approx n^2$

How to $\boxed{\text{slow down reaction } L_2 \rightarrow L_1}$?

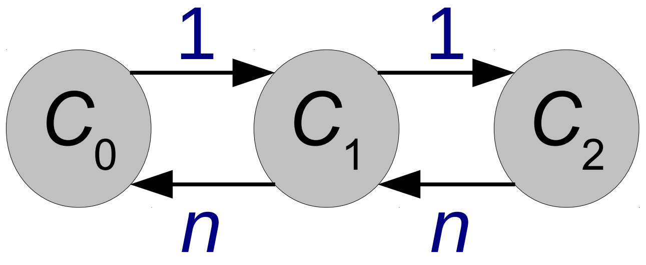
Use a clock:

$$\boxed{C_2 +} L_2 \rightarrow \boxed{C_0 +} L_1$$

$1\ C_0$, $1\ F$, $n\ B$

$F + C_0 \rightarrow F + C_1$ $B + C_1 \rightarrow B + C_0$

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



reverse-biased random walk

C_2 appears after expected time $\approx n^2$

How to $\boxed{\text{slow down reaction } L_2 \to L_1}$?

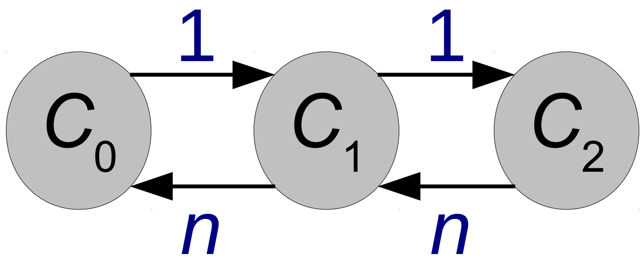
Use a clock:

$$\boxed{C_2 +} L_2 \to \boxed{C_0 +} L_1$$

$1\ C_0$, $1\ F$, $n\ B$

$F + C_0 \to F + C_1$ $B + C_1 \to B + C_0$

$F + C_1 \to F + C_2$ $B + C_2 \to B + C_1$


$$\begin{tikzcd} C_0 \arrow[r, bend left, "1"] & C_1 \arrow[l, bend left, "n"] \arrow[r, bend left, "1"] & C_2 \arrow[l, bend left, "n"] \end{tikzcd}$$

reverse-biased random walk

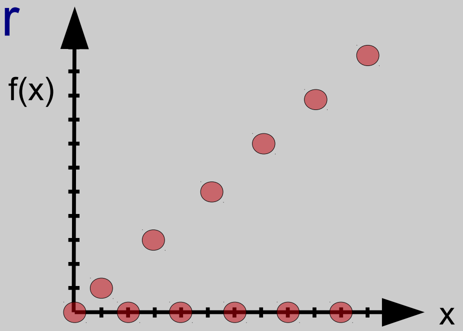
C_2 appears after expected time $\approx n^2$

E[time for $L_2 + R \to L_3$] $\leq n$

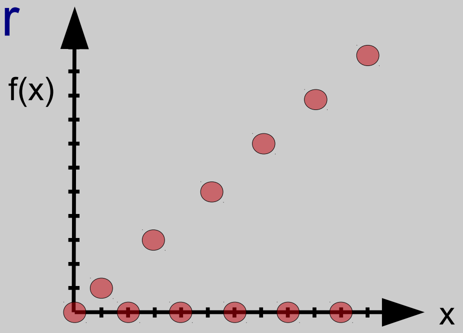
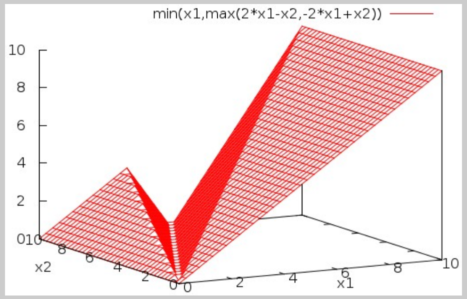
Summary

	integer-valued (stochastic) CRN	real-valued (mass-action) CRN
rate independent		
rate dependent		

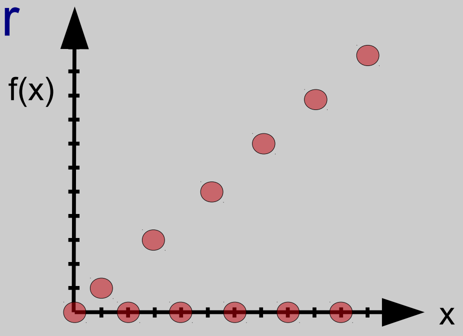
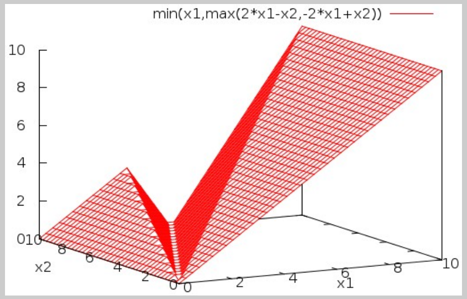
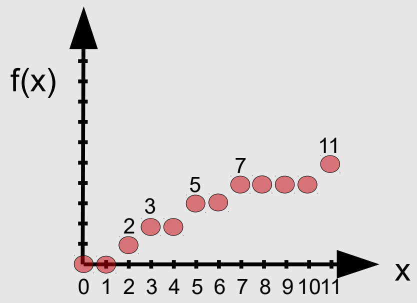
Summary

	integer-valued (stochastic) CRN	real-valued (mass-action) CRN
rate independent	semilinear  [Chen, D, Soloveichik, *DNA 2012, NaCo 2013*] [Angluin, Aspnes, Eisenstat, *PODC 2006*]	
rate dependent		

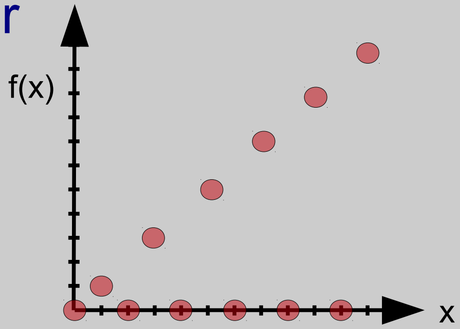
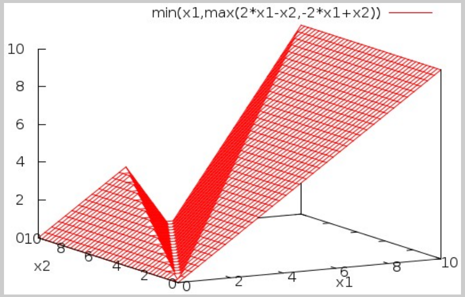
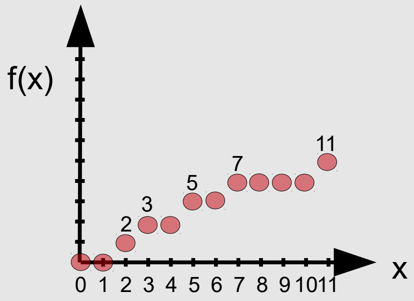
Summary

	integer-valued (stochastic) CRN	real-valued (mass-action) CRN
rate independent	semilinear  [Chen, D, Soloveichik, *DNA 2012, NaCo 2013*] [Angluin, Aspnes, Eisenstat, *PODC 2006*]	continuous piecewise linear  [Chen, D, Soloveichik, *ITCS 2014*]
rate dependent		

Summary

	integer-valued (stochastic) CRN	**real-valued (mass-action) CRN**
rate independent	semilinear  [Chen, D, Soloveichik, *DNA 2012, NaCo 2013*] [Angluin, Aspnes, Eisenstat, *PODC 2006*]	continuous piecewise linear  [Chen, D, Soloveichik, *ITCS 2014*]
rate dependent	Turing computable  [Soloveichik, Cook, Winfree, Bruck, *NaCo 2008*] [Angluin, Aspnes, Eisenstat, *DISC 2006*]	

Summary

	integer-valued (stochastic) CRN	real-valued (mass-action) CRN
rate independent	semilinear  [Chen, D, Soloveichik, *DNA 2012, NaCo 2013*] [Angluin, Aspnes, Eisenstat, *PODC 2006*]	continuous piecewise linear  [Chen, D, Soloveichik, *ITCS 2014*]
rate dependent	Turing computable  [Soloveichik, Cook, Winfree, Bruck, *NaCo 2008*] [Angluin, Aspnes, Eisenstat, *DISC 2006*]	???

Conclusion

Conclusion

- chemical reactions can be programmed to process information encoded in counts/concentrations

Conclusion

- chemical reactions can be programmed to process information encoded in counts/concentrations
  - error-free computation is limited in ability (*piecewise linear functions*)

Conclusion

- chemical reactions can be programmed to process information encoded in counts/concentrations
  - error-free computation is limited in ability (*piecewise linear functions*)
  - allowing small probability of error **vastly** expands ability (*all computable functions*)

Conclusion

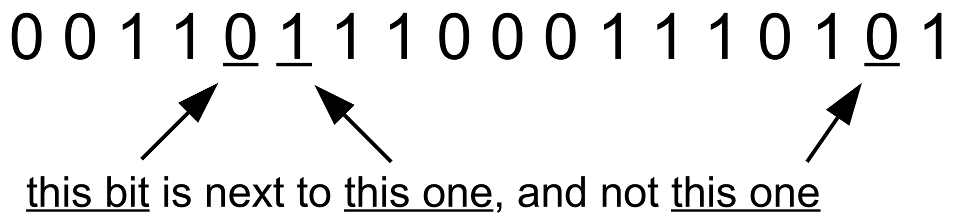
- chemical reactions can be programmed to process information encoded in counts/concentrations
  - error-free computation is limited in ability (*piecewise linear functions*)
  - allowing small probability of error **vastly** expands ability (*all computable functions*)

- key challenge is representing information without geometry

Conclusion

- chemical reactions can be programmed to process information encoded in counts/concentrations
  - error-free computation is limited in ability (*piecewise linear functions*)
  - allowing small probability of error **vastly** expands ability (*all computable functions*)

- key challenge is representing information without geometry



Acknowledgments

Ho-Lin Chen

Rachel Cummings

David Soloveichik

NSF · Computing Research Association (CRA) · Computing Community Consortium (CCC)

American Institute of Mathematics

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