Computation by (not about) chemistry

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

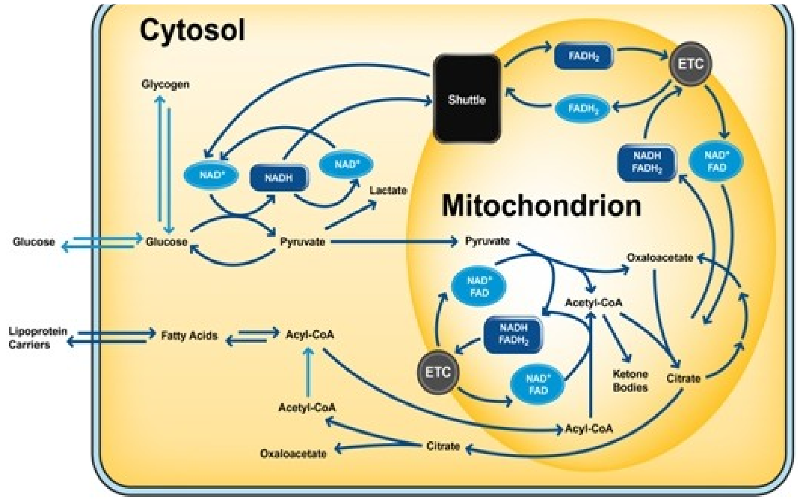
David Doty

The software of life

How does the cell compute?

The software of life

How does the cell compute?

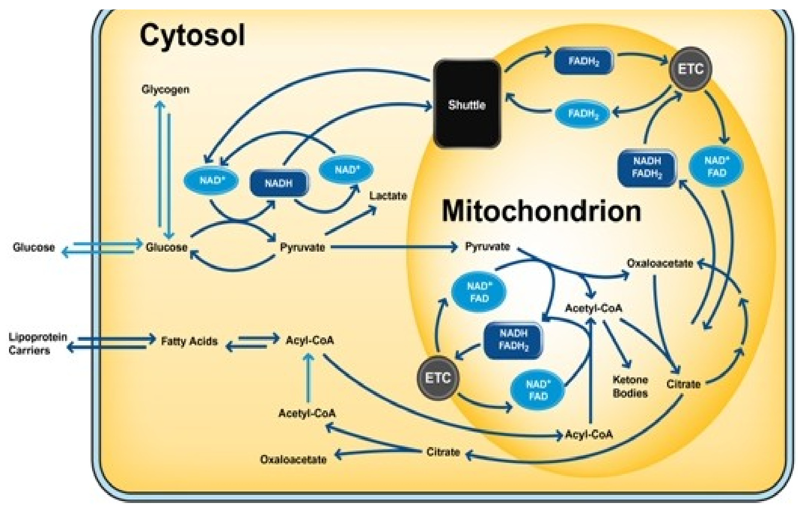


chemistry / geometry

The software of life

~~How does the cell compute?~~

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



Chemical reaction networks (CRN)

Chemical reaction networks (CRN)

$$R \to P_1 + P_2$$

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)

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

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

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

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

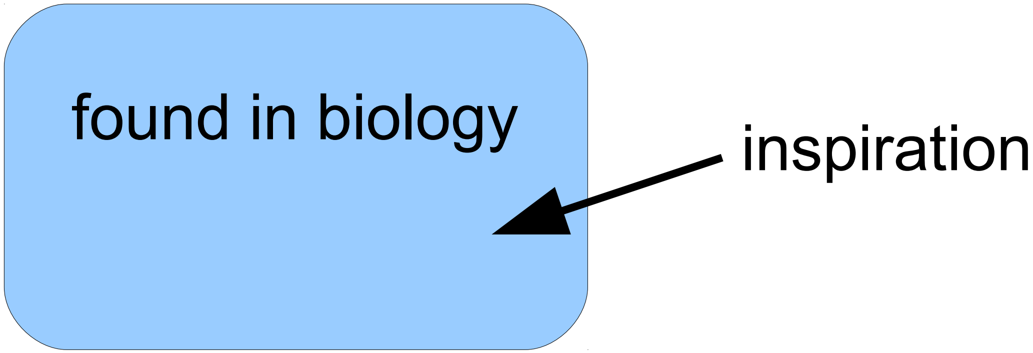
(anonymous
waste product)

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

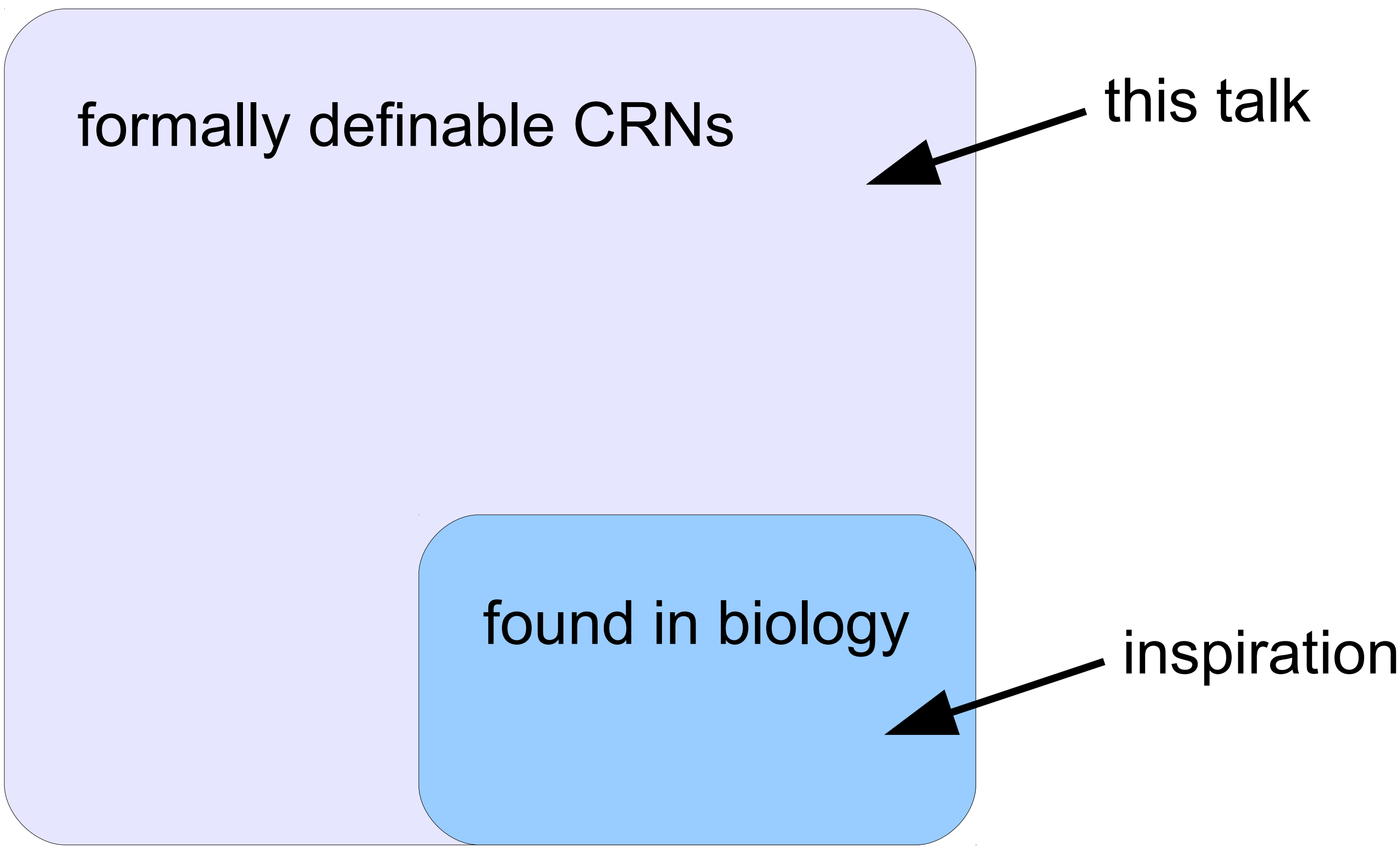
(anonymous
fuel source)

What behavior is possible for chemistry in principle?

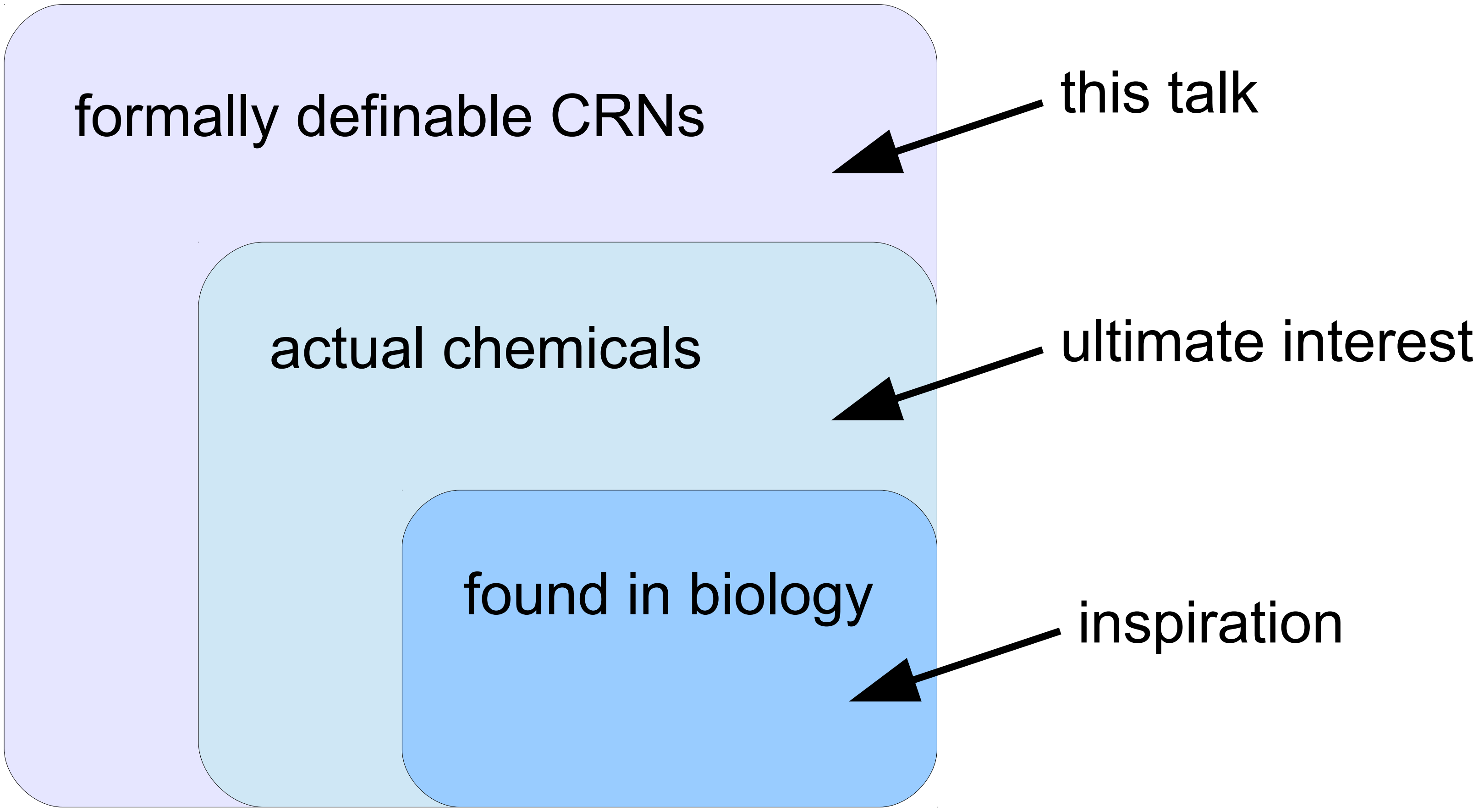
What behavior is possible for chemistry in principle?



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*

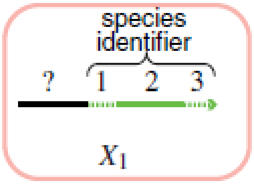
$$X_1 + X_2 \to X_3$$

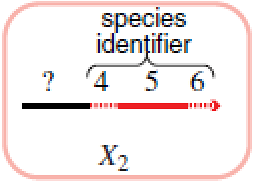
Can we compute with chemistry?

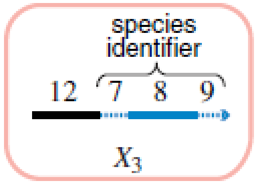
"Not every crazy CRN you scribble on paper describes actual chemicals!"

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

$$X_1 + X_2 \rightarrow X_3$$





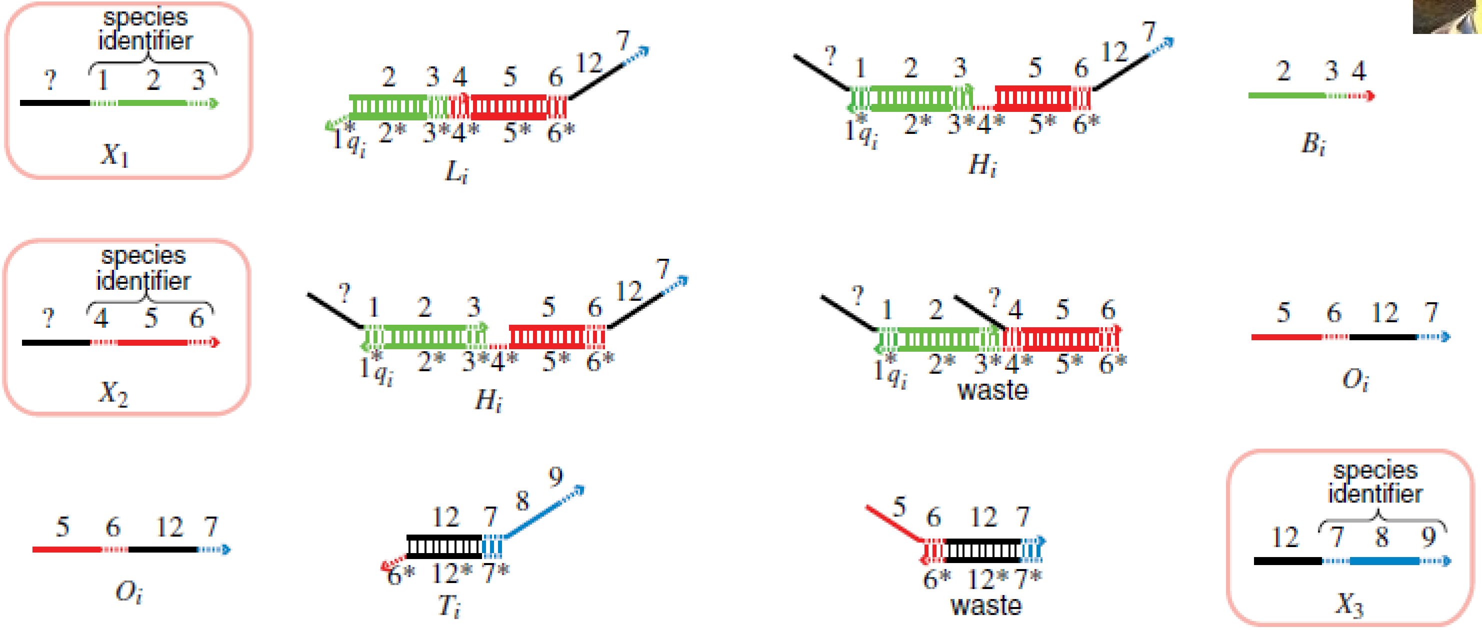


Can we compute with chemistry?

"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*

$$X_1 + X_2 \rightarrow X_3$$

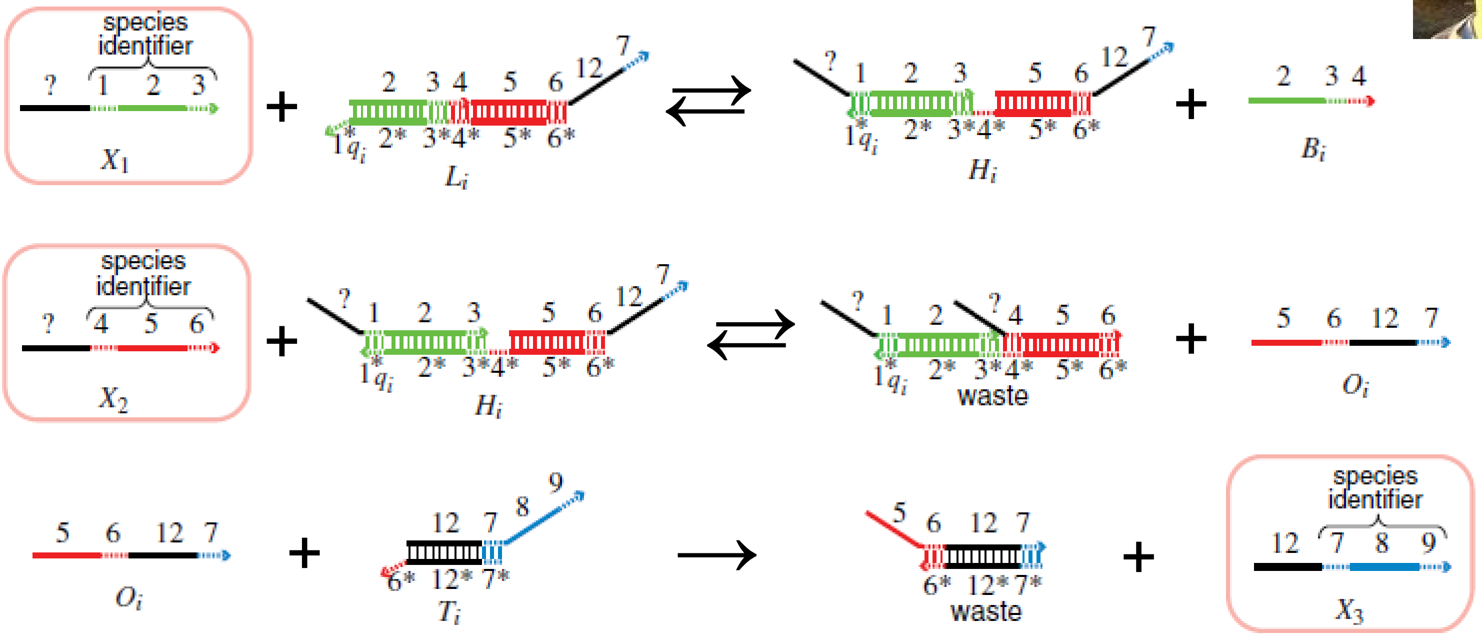


Can we compute with chemistry?

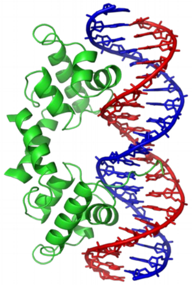
“Not every crazy CRN you scribble on paper describes actual chemicals!”

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

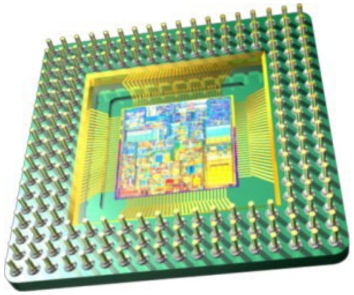
$$X_1 + X_2 \rightarrow X_3$$



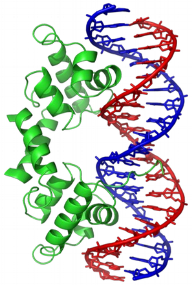
Why compute with chemistry?



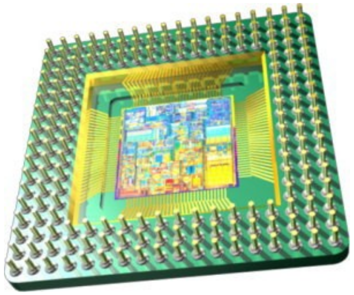
versus



Why compute with chemistry?

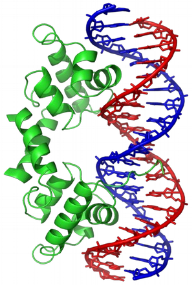


versus

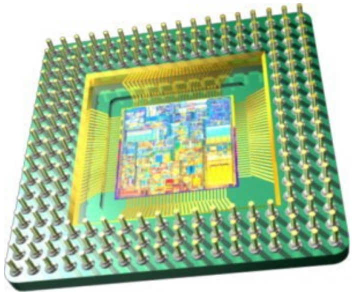


speed?

Why compute with chemistry?



versus

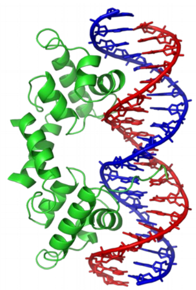


slower

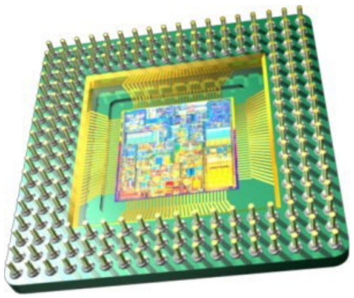
speed?

faster

Why compute with chemistry?



versus

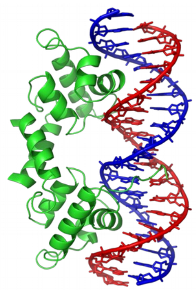


slower

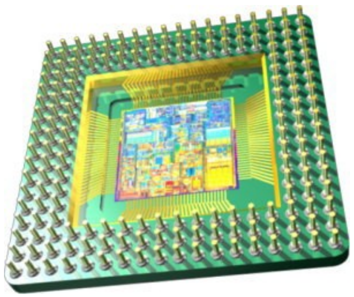
~~speed?~~

faster

Why compute with chemistry?



versus



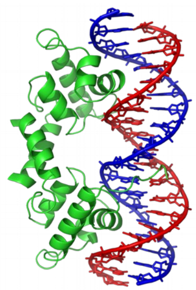
slower

~~speed?~~

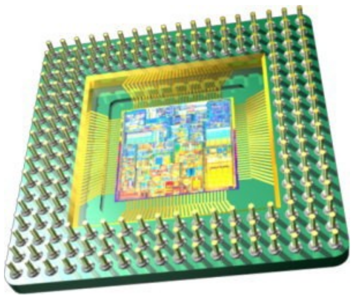
faster

component size?

Why compute with chemistry?



versus



slower

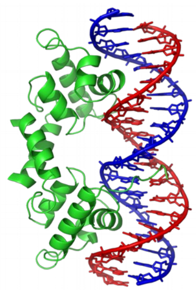
~~speed?~~

faster

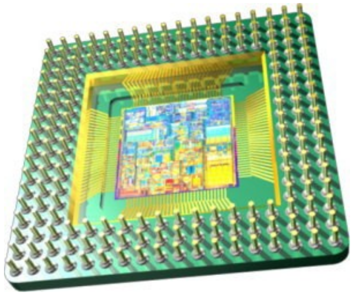
≈ 10-100 nm

component size?

Why compute with chemistry?

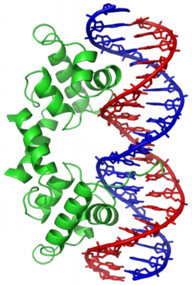


versus

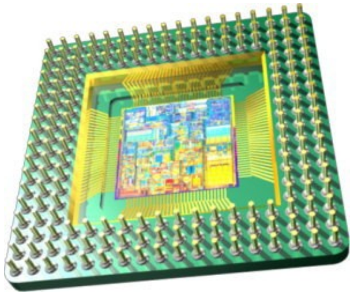


slower	~~speed?~~	faster
≈ 10-100 nm	component size?	≈ 10-100 nm

Why compute with chemistry?

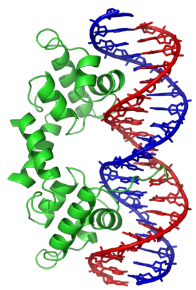


versus

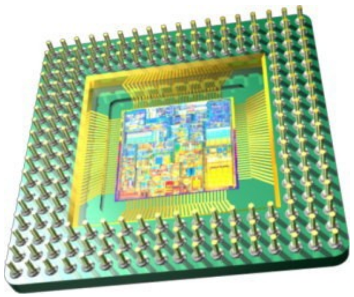


slower	~~speed?~~	faster
≈ 10-100 nm	~~component size?~~	≈ 10-100 nm

Why compute with chemistry?

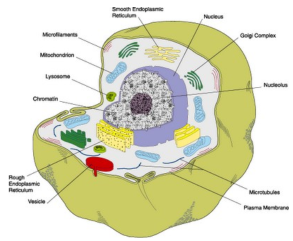


versus



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

cells

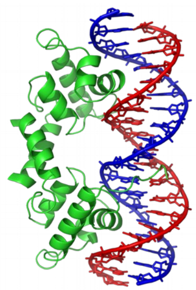


“smart drug” released only in certain cellular conditions

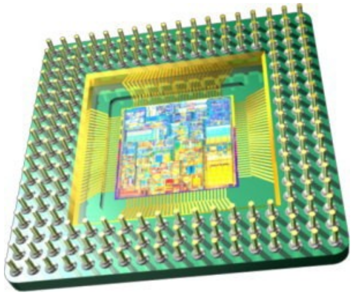
bioreactors

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

Why compute with chemistry?

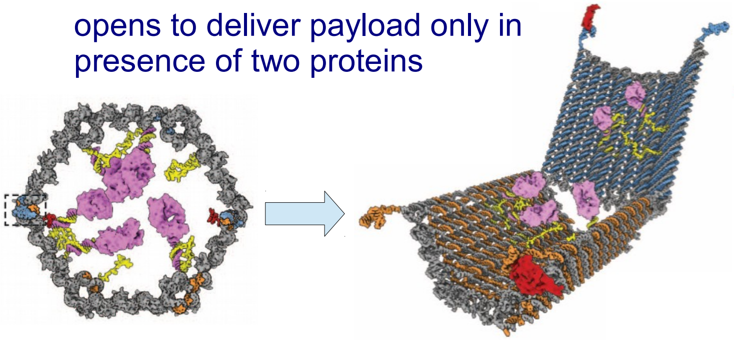


versus



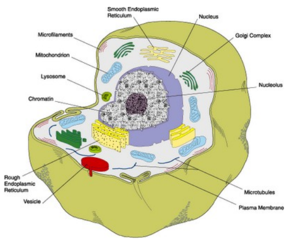
slower	~~speed?~~	faster
≈ 10-100 nm	~~component size?~~	≈ 10-100 nm
yes	Compatible with biological or other “wet environments”?	not easily

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.

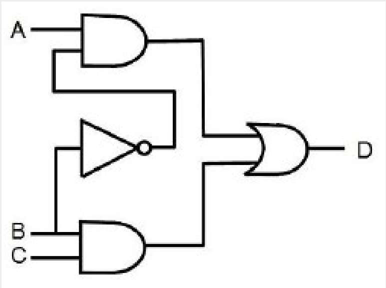
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

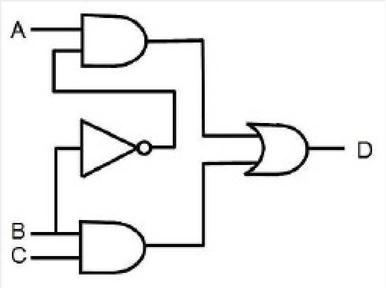
Boolean logic



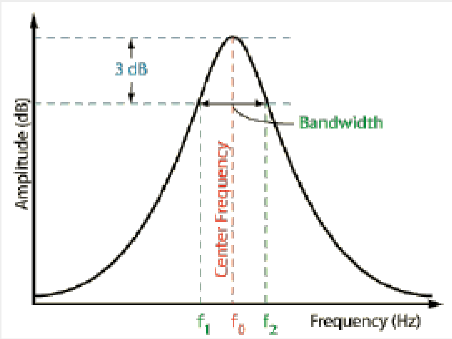
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

Boolean logic



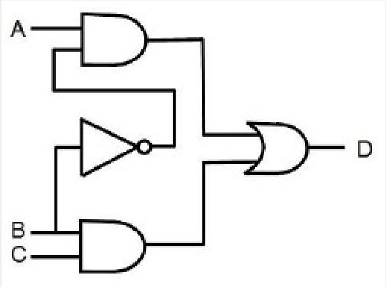
signal processing



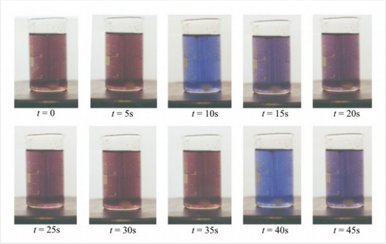
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

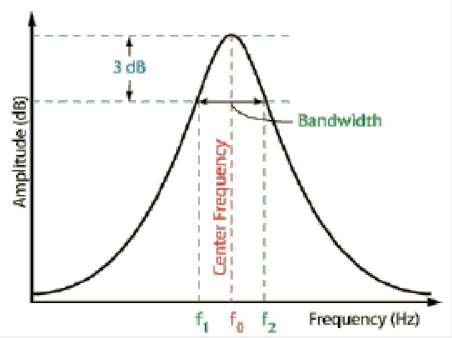
Boolean logic



oscillation



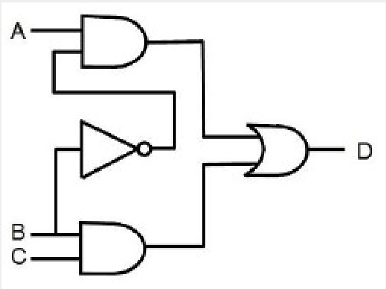
signal processing



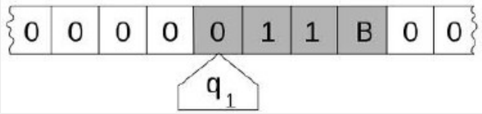
What does it mean to compute with chemistry?

CRNs have a wide range of behaviors:

Boolean logic

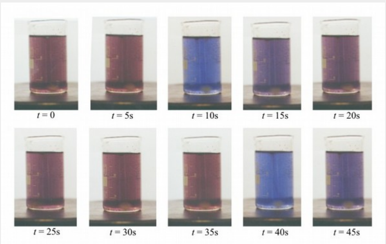


discrete algorithms

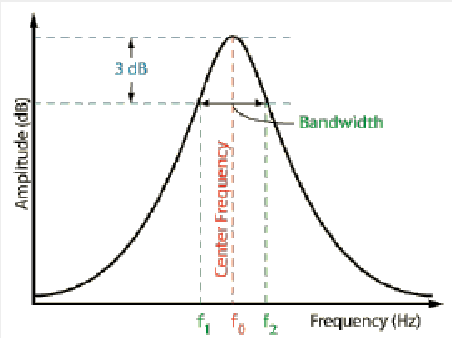


analog computing

oscillation



signal processing



Integer-valued kinetic CRN model

Integer-valued kinetic CRN model

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

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, …}
- **reactions**:

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

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

- **state**: integer vector of *counts* **s** = (#X, #Y, ...)

Integer-valued kinetic CRN model

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

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

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

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

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•#X
$A + B \xrightarrow{k_2} X$	k_2•#A•#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: $f(x) = x/2$

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

input species: X

output species: Y

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

CRN function computation (example)

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

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

input species: X

output species: Y

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

#Y = $x/2$ expected at equilibrium

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

---

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 = $\cancel{x/2}$ $x/3$ expected at equilibrium

---

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

#Y stabilizes, with expected value $\cancel{x/2}$ $x/3$

Rate-independent CRN computation

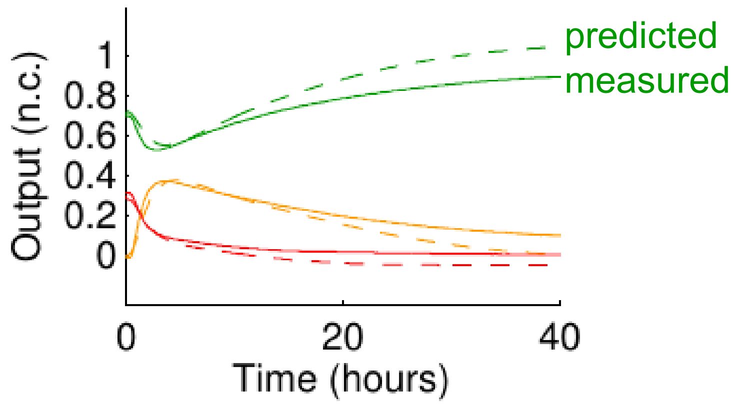
What can CRNs compute when we
don't know/can't control the rates?

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

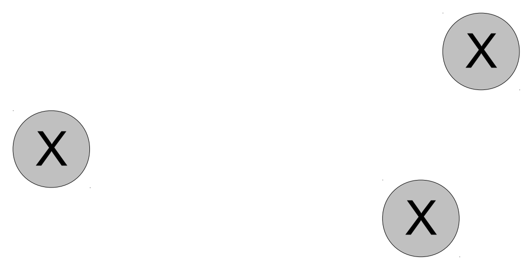
CRN function computation (example)

function: $f(x) = 2x$

input species: X

output species: Y

reactions: ??

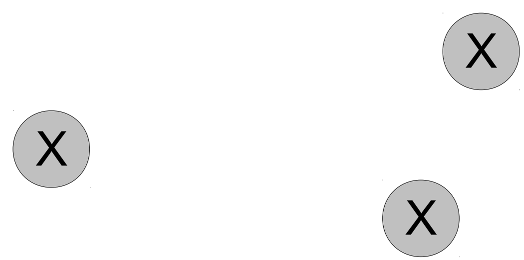


CRN function computation (example)

function: $f(x) = 2x$

input species: X
output species: Y

reactions: $X \to 2Y$

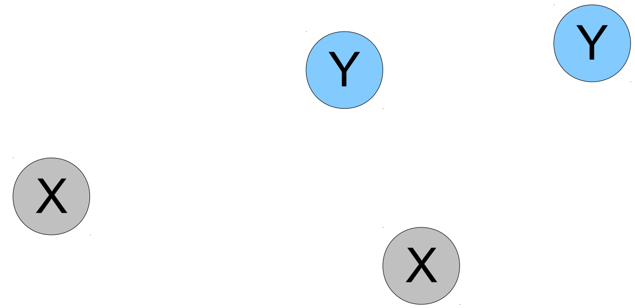


CRN function computation (example)

function: $f(x) = 2x$

input species: X
output species: Y

reactions: $X \rightarrow 2Y$

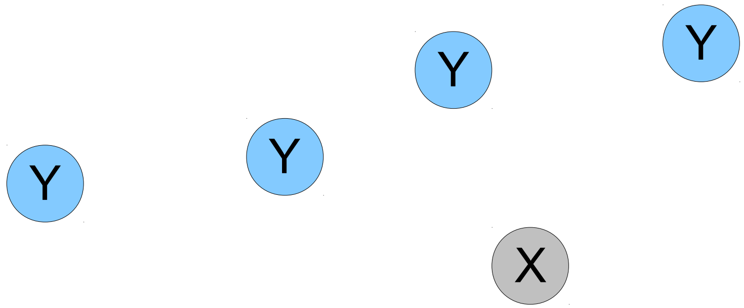


CRN function computation (example)

function: $f(x) = 2x$

input species: X
output species: Y

reactions: $X \rightarrow 2Y$

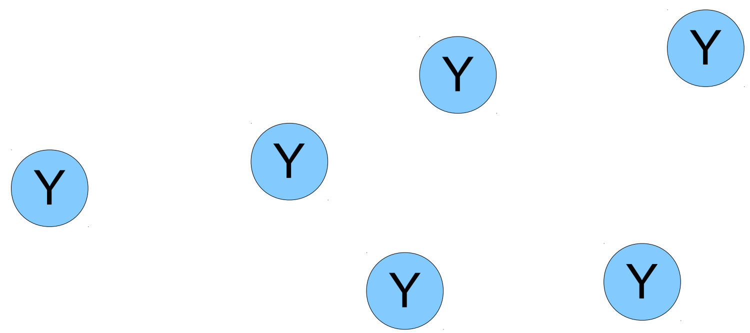


CRN function computation (example)

function: $f(x) = 2x$

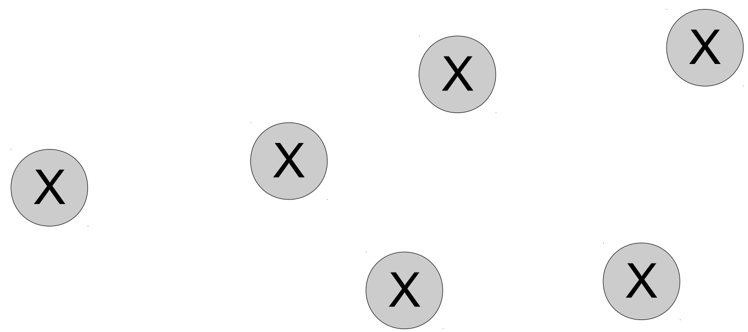
input species: X
output species: Y

reactions: $X \rightarrow 2Y$



CRN function computation (example)

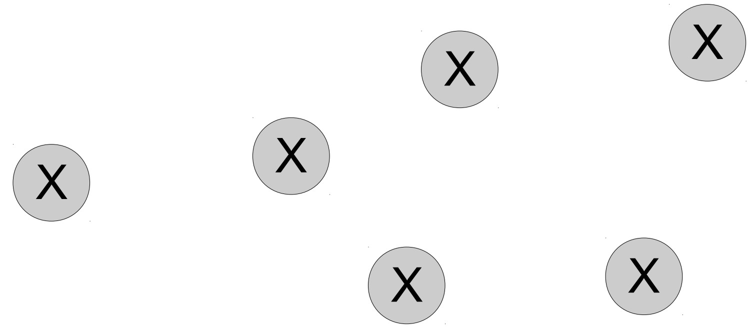
function: $f(x) = x/2$



CRN function computation (example)

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

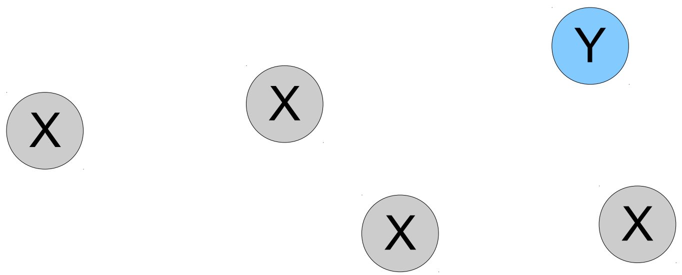
reactions: $2X \rightarrow Y$



CRN function computation (example)

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

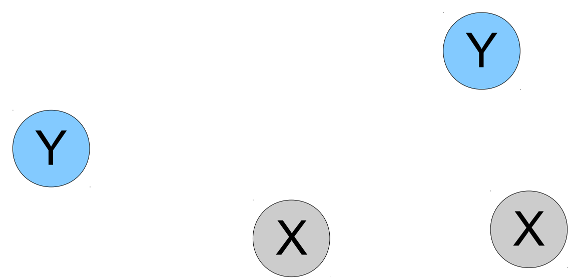
reactions: $2X \rightarrow Y$



CRN function computation (example)

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

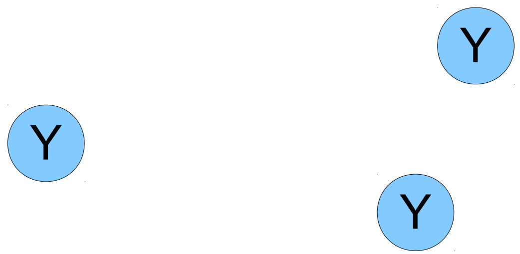
reactions: $2X \rightarrow Y$



CRN function computation (example)

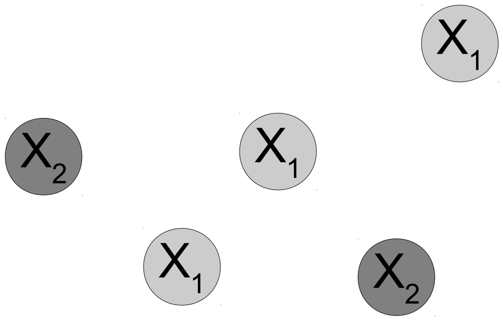
function: $f(x) = x/2$

reactions: $2X \rightarrow Y$



CRN function computation (example)

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

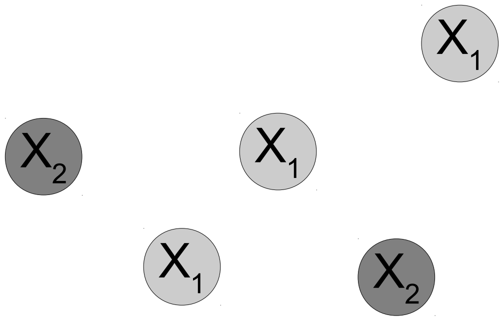


CRN function computation (example)

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

reactions: $X_1 \rightarrow Y$

$X_2 \rightarrow Y$

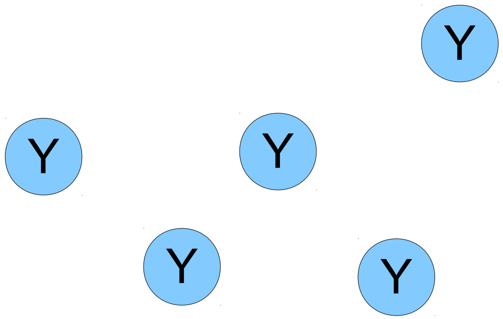


CRN function computation (example)

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

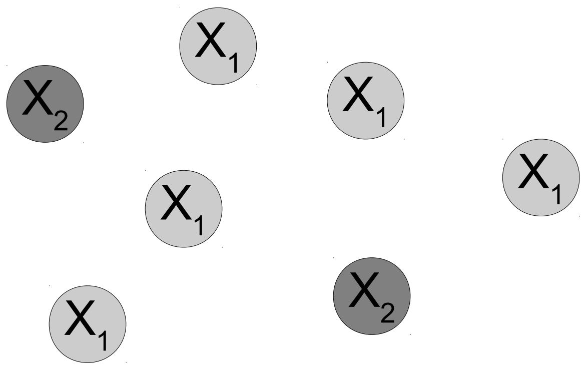
reactions: $X_1 \rightarrow Y$

$X_2 \rightarrow Y$



CRN function computation (example)

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

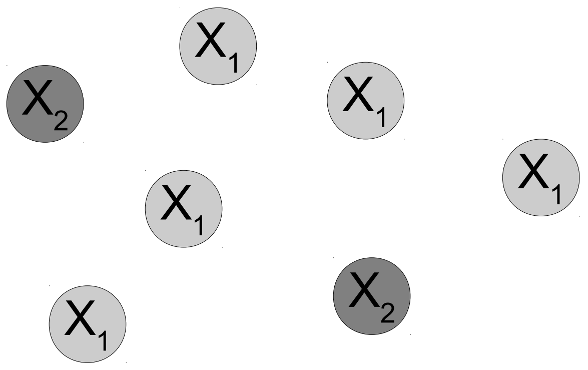


CRN function computation (example)

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

reactions: $X_1 \rightarrow Y$

$X_2 + Y \rightarrow$

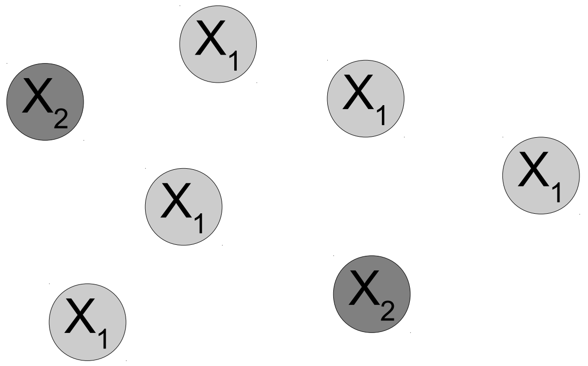


CRN function computation (example)

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

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

$X_2 + Y \to$

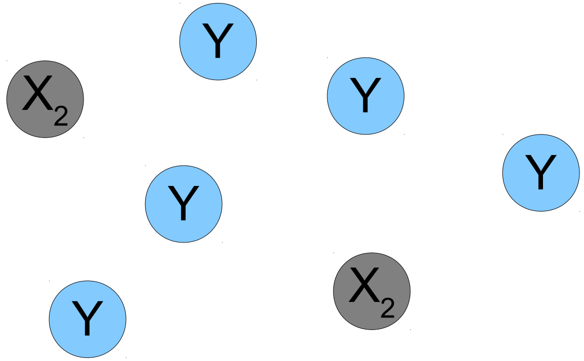


CRN function computation (example)

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

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

$X_2 + Y \rightarrow$

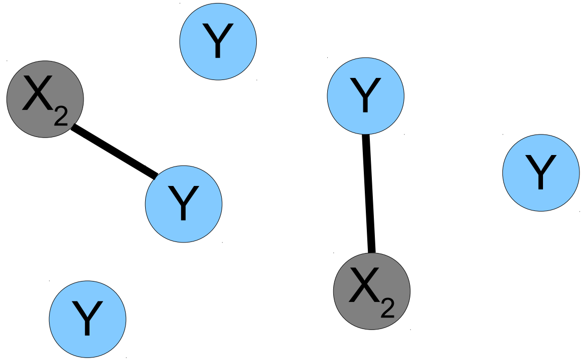


CRN function computation (example)

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

reactions: $X_1 \rightarrow Y$

$\boxed{X_2 + Y \rightarrow}$

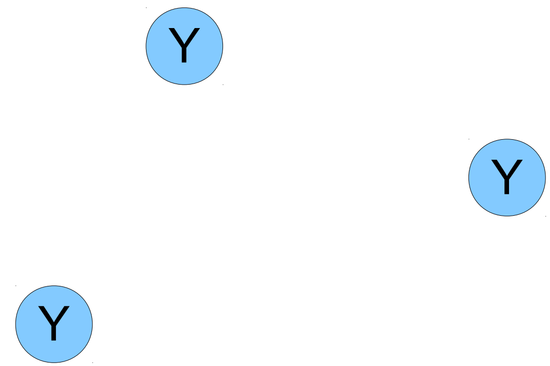


CRN function computation (example)

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

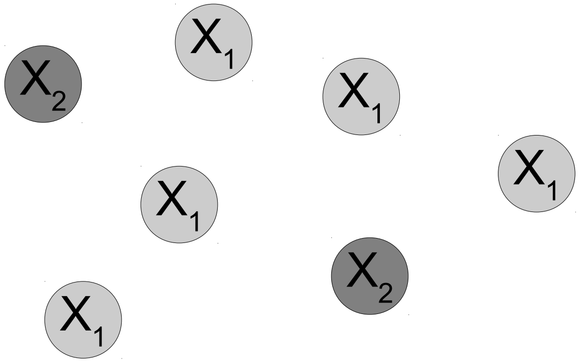
reactions: $X_1 \rightarrow Y$

$X_2 + Y \rightarrow$



CRN function computation (example)

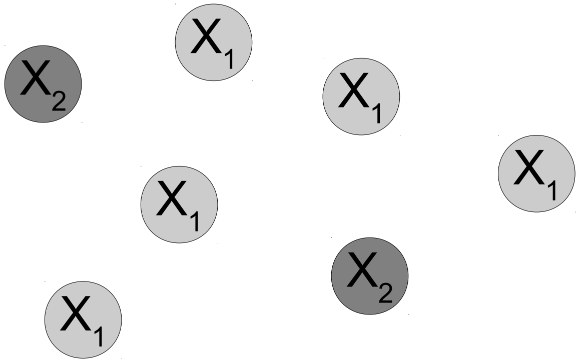
function: $f(x_1,x_2) = \min\{x_1,x_2\}$



CRN function computation (example)

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

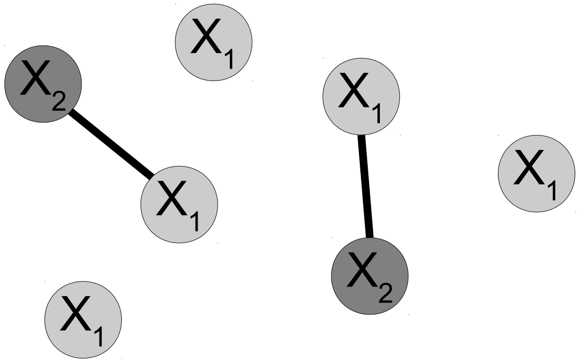
reactions: $X_1 + X_2 \rightarrow Y$



CRN function computation (example)

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

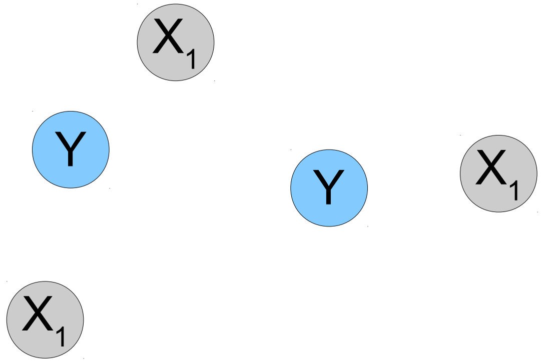
reactions: $X_1 + X_2 \rightarrow Y$



CRN function computation (example)

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

reactions: $X_1 + X_2 \rightarrow Y$

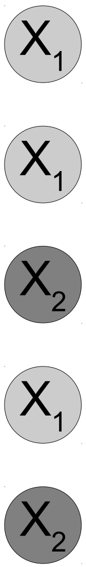


CRN function computation (example)

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

CRN function computation (example)

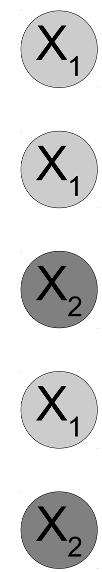
function: $f(x_1,x_2) = \max\{x_1,x_2\} = x_1 + x_2 - \min\{x_1,x_2\}$



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

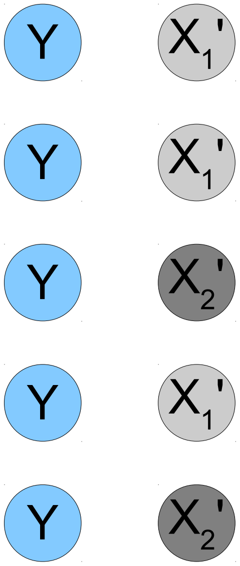


CRN function computation (example)

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

reactions:

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



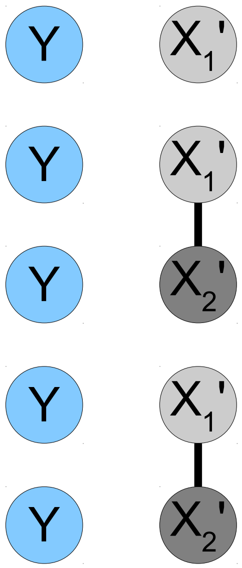
CRN function computation (example)

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

reactions: $X_1 \to Y + X_1'$

$X_2 \to Y + X_2'$

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



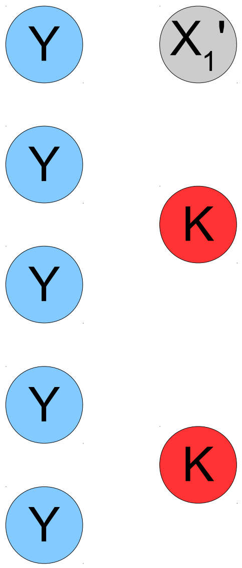
CRN function computation (example)

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

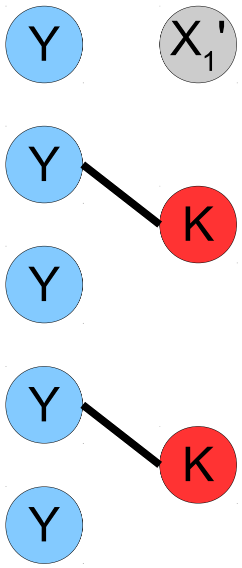
reactions:

$$X_1 \to Y + X_1'$$

$$X_2 \to Y + X_2'$$

$$X_1' + X_2' \to K$$

$$\boxed{K + Y \to }$$

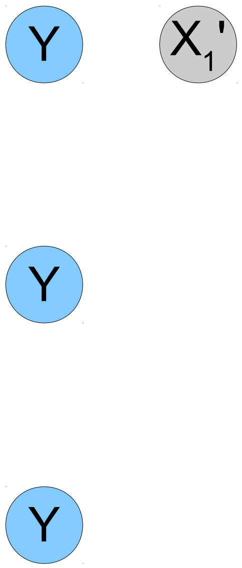


CRN function computation (example)

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

reactions:

$$X_1 \rightarrow Y + X_1'$$
$$X_2 \rightarrow Y + X_2'$$
$$X_1' + X_2' \rightarrow K$$
$$K + Y \rightarrow$$



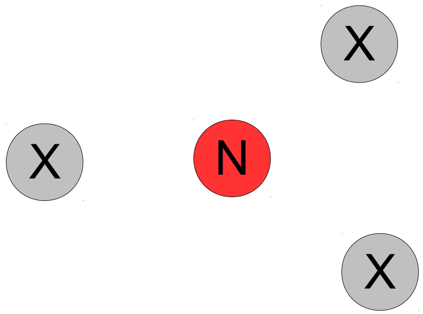
CRN predicate computation (example)

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

CRN predicate computation (example)

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

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

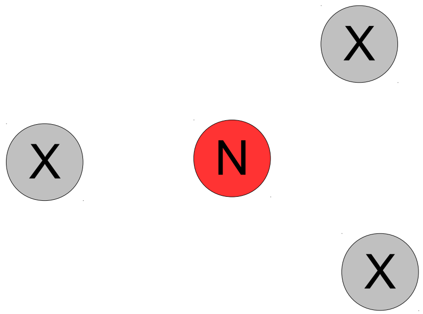


CRN predicate computation (example)

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

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

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



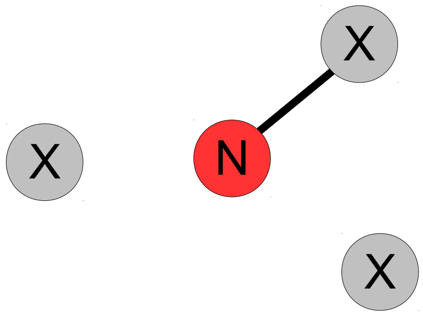
CRN predicate computation (example)

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

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

reactions: $\boxed{N + X \rightarrow Y}$

$Y + X \rightarrow N$

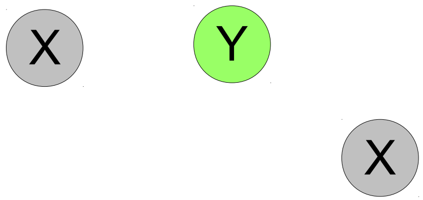


CRN predicate computation (example)

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

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

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



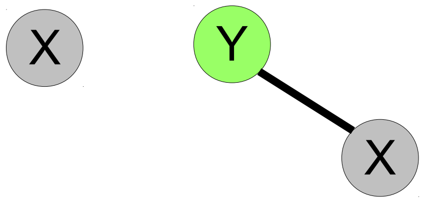
CRN predicate computation (example)

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

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

reactions: $N + X \rightarrow Y$

$\boxed{Y + X \rightarrow N}$



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” $\Leftrightarrow x$ is odd)

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

reactions: $\boxed{N + X \rightarrow Y}$

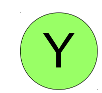
$Y + X \rightarrow N$

CRN predicate computation (example)

predicate: $p(x)$: parity of x (“yes” $\Leftrightarrow x$ is odd)

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

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



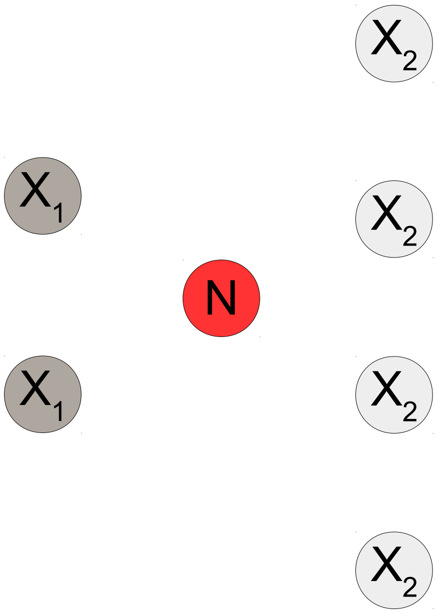
CRN predicate computation (example)

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



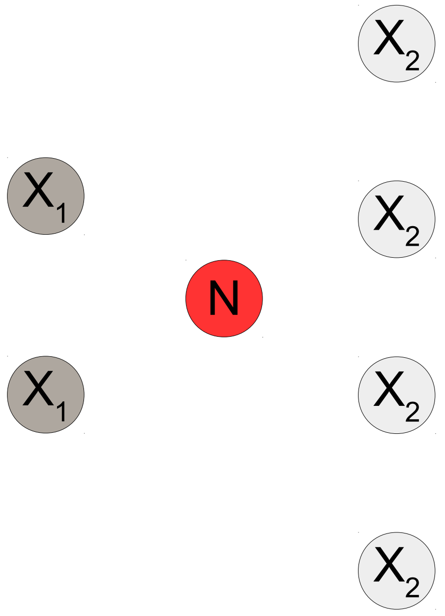
CRN predicate computation (example)

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

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

reactions: $N + X_1 \rightarrow Y$

$Y + X_2 \rightarrow N$



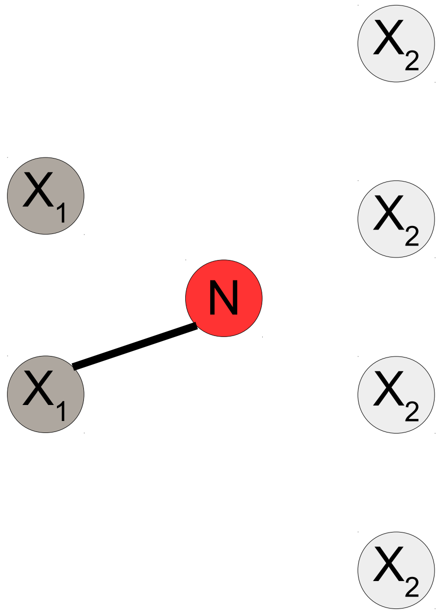
CRN predicate computation (example)

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

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

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

$Y + X_2 \rightarrow N$



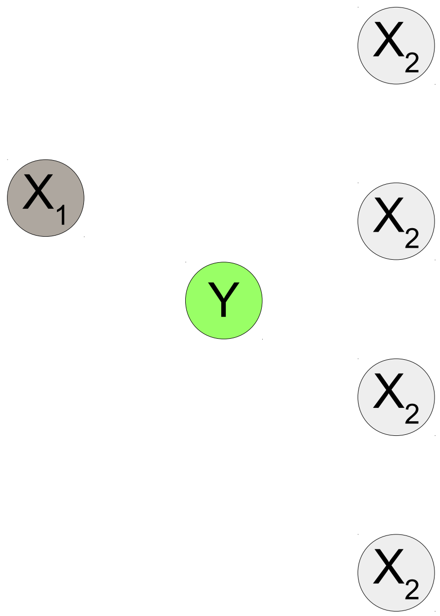
CRN predicate computation (example)

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

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

reactions: $N + X_1 \rightarrow Y$

$Y + X_2 \rightarrow N$



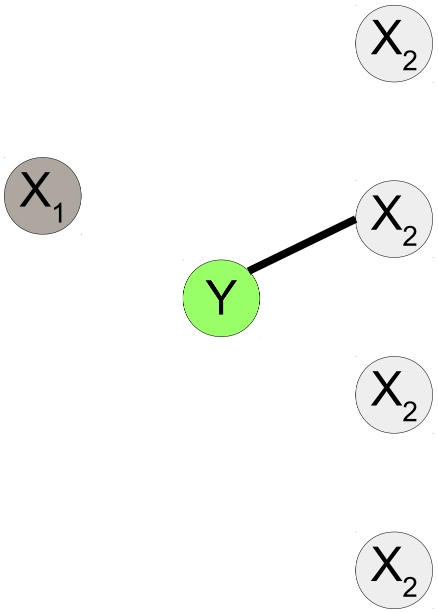
CRN predicate computation (example)

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

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

reactions: $N + X_1 \rightarrow Y$

$\boxed{Y + X_2 \rightarrow N}$



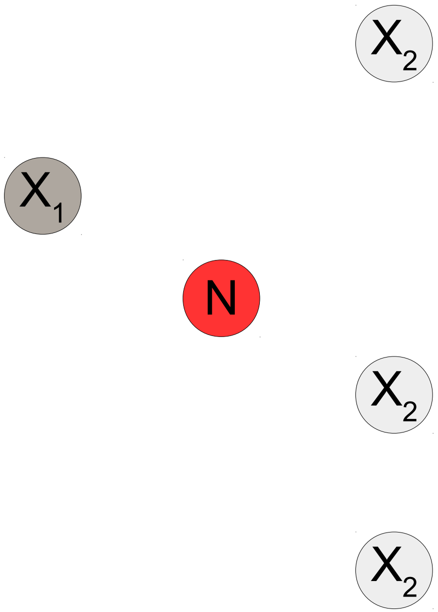
CRN predicate computation (example)

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

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

reactions: $N + X_1 \rightarrow Y$

$Y + X_2 \rightarrow N$



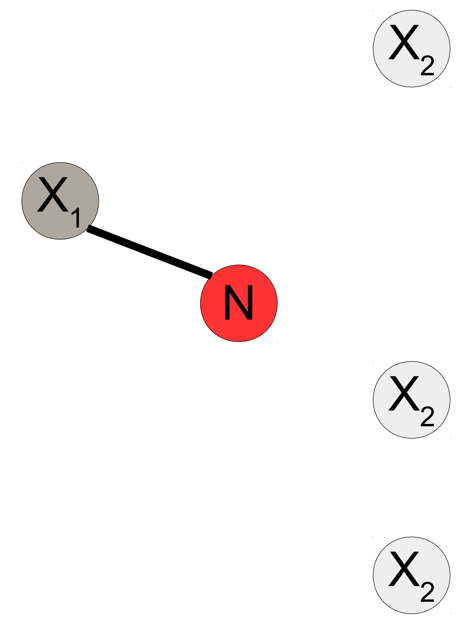
CRN predicate computation (example)

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

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

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

$Y + X_2 \rightarrow N$



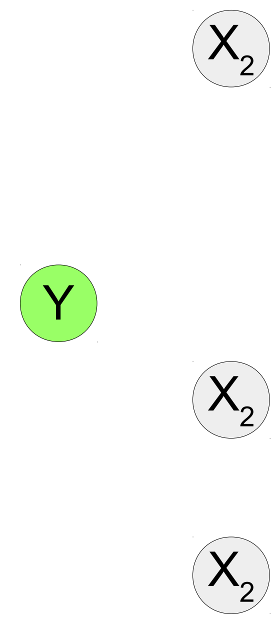
CRN predicate computation (example)

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

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

reactions: $N + X_1 \rightarrow Y$

$Y + X_2 \rightarrow N$



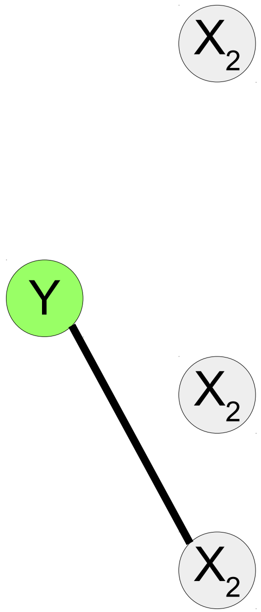
CRN predicate computation (example)

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

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

reactions: $N + X_1 \rightarrow Y$

$\boxed{Y + X_2 \rightarrow N}$

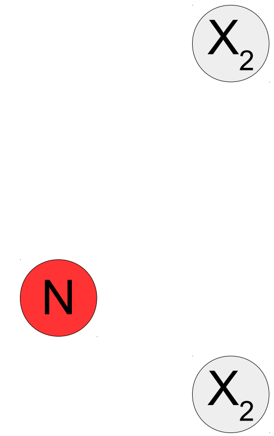


CRN predicate computation (example)

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

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

reactions:
$$N + X_1 \to Y$$
$$Y + X_2 \to N$$



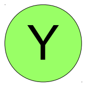
CRN predicate computation (example)

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

CRN predicate computation (example)

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

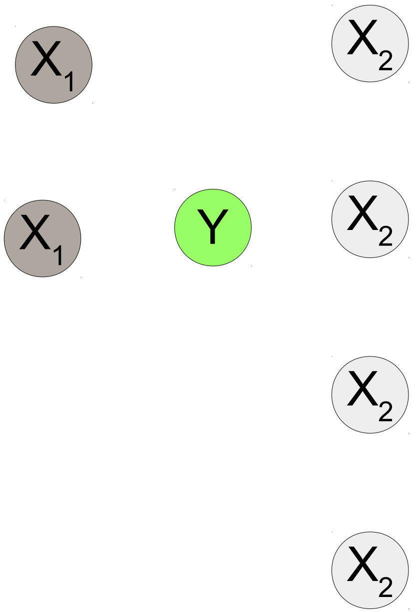
initial state: $\{\ x_1\ X_1\ ,\ x_2\ X_2\ ,\ 1\ Y\ \}$



CRN predicate computation (example)

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

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



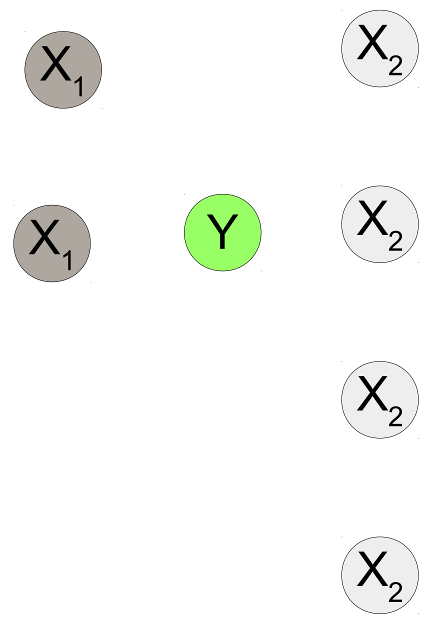
CRN predicate computation (example)

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

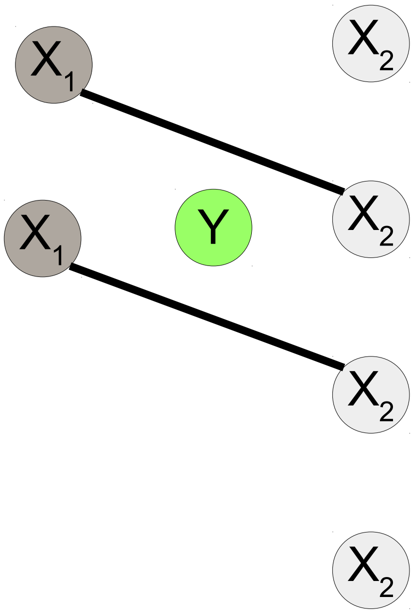
initial state: $\{\, x_1\, X_1\,,\ x_2\, X_2\,,\ 1\ Y \,\}$

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

$Y + N \rightarrow Y$

$X_1 + Y \rightarrow X_1 + N$

$X_2 + Y \rightarrow X_2 + N$



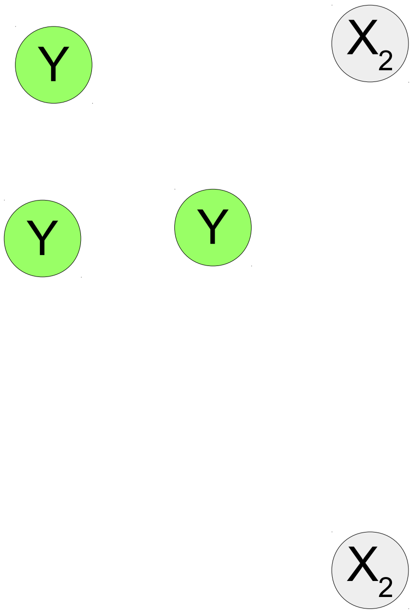
CRN predicate computation (example)

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

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

reactions:

$$X_1 + X_2 \rightarrow Y$$
$$Y + N \rightarrow Y$$
$$X_1 + Y \rightarrow X_1 + N$$
$$X_2 + Y \rightarrow X_2 + N$$

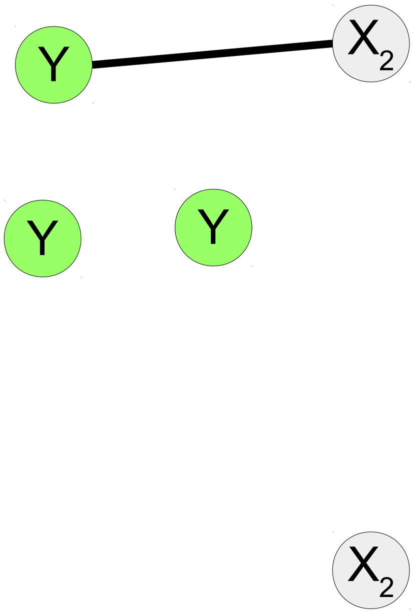


CRN predicate computation (example)

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

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

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



CRN predicate computation (example)

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

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

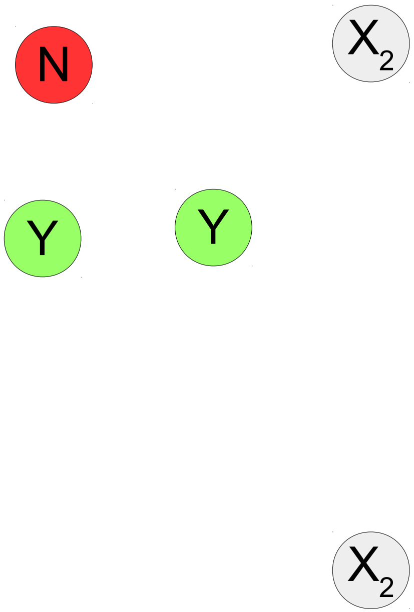
reactions:

$$X_1 + X_2 \rightarrow Y$$

$$Y + N \rightarrow Y$$

$$X_1 + Y \rightarrow X_1 + N$$

$$\boxed{X_2 + Y \rightarrow X_2 + N}$$



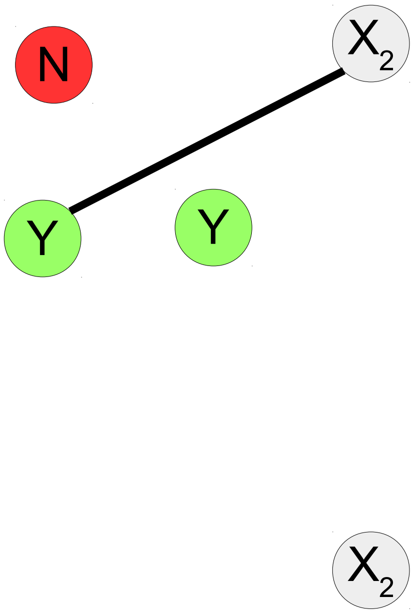
CRN predicate computation (example)

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

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

reactions:

$$X_1 + X_2 \rightarrow Y$$
$$Y + N \rightarrow Y$$
$$X_1 + Y \rightarrow X_1 + N$$
$$\boxed{X_2 + Y \rightarrow X_2 + N}$$



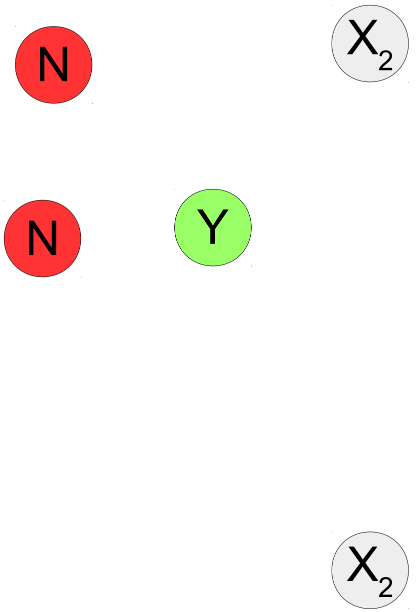
CRN predicate computation (example)

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

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

reactions:

$$X_1 + X_2 \rightarrow Y$$
$$Y + N \rightarrow Y$$
$$X_1 + Y \rightarrow X_1 + N$$
$$\boxed{X_2 + Y \rightarrow X_2 + N}$$



CRN predicate computation (example)

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

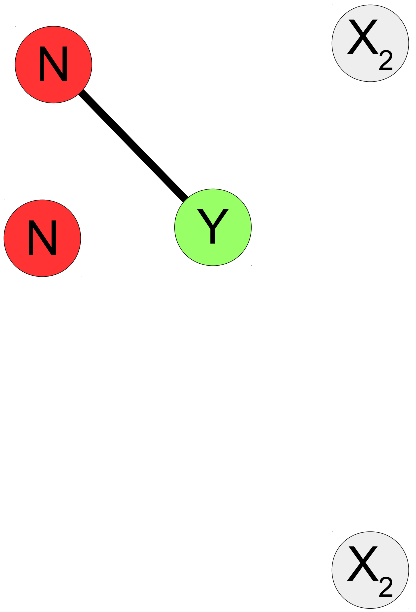
initial state: $\{ x_1\, X_1 \,,\, x_2\, X_2 \,,\, 1\ Y \}$

reactions: $X_1 + X_2 \to Y$

$\boxed{Y + N \to Y}$

$X_1 + Y \to X_1 + N$

$X_2 + Y \to X_2 + N$



CRN predicate computation (example)

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

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

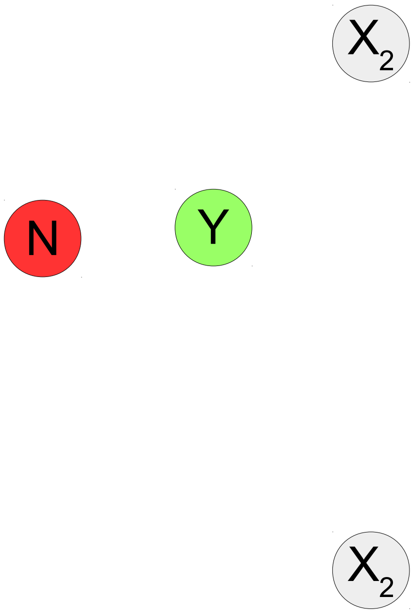
reactions:

$X_1 + X_2 \rightarrow Y$

$\boxed{Y + N \rightarrow Y}$

$X_1 + Y \rightarrow X_1 + N$

$X_2 + Y \rightarrow X_2 + N$



CRN predicate computation (example)

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

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

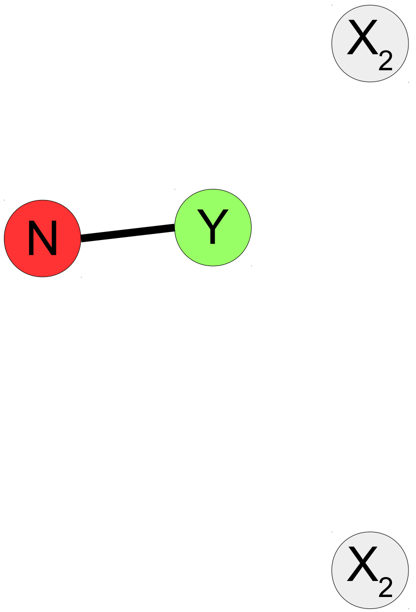
reactions:

$$X_1 + X_2 \rightarrow Y$$

$$Y + N \rightarrow Y$$

$$X_1 + Y \rightarrow X_1 + N$$

$$X_2 + Y \rightarrow X_2 + N$$



CRN predicate computation (example)

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

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

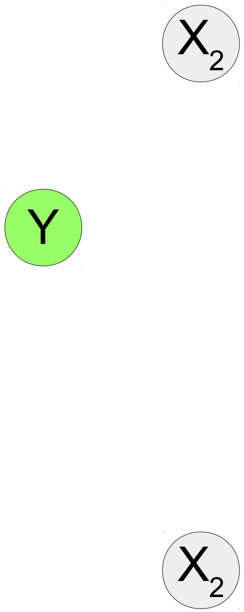
reactions:

$$X_1 + X_2 \rightarrow Y$$

$$\boxed{Y + N \rightarrow Y}$$

$$X_1 + Y \rightarrow X_1 + N$$

$$X_2 + Y \rightarrow X_2 + N$$



CRN predicate computation (example)

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

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

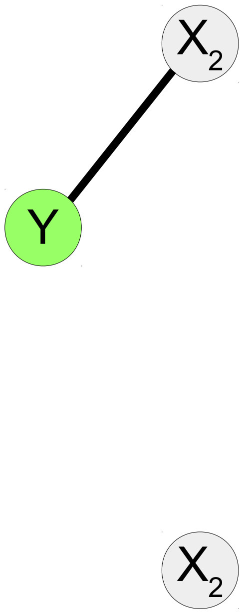
reactions:

$$X_1 + X_2 \to Y$$

$$Y + N \to Y$$

$$X_1 + Y \to X_1 + N$$

$$\boxed{X_2 + Y \to X_2 + N}$$



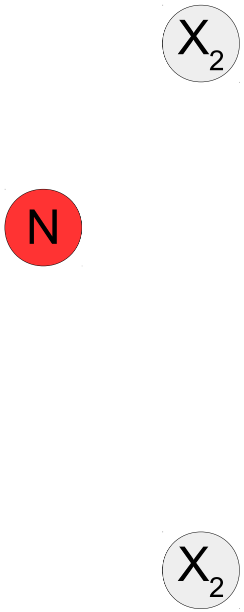
CRN predicate computation (example)

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

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

reactions:

$$X_1 + X_2 \rightarrow Y$$
$$Y + N \rightarrow Y$$
$$X_1 + Y \rightarrow X_1 + N$$
$$X_2 + Y \rightarrow X_2 + N$$



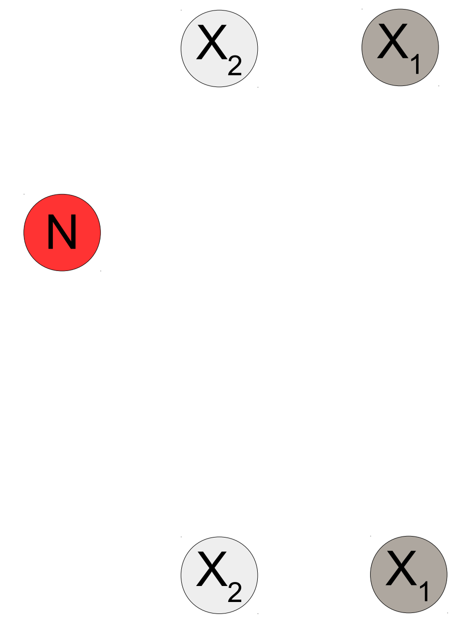
CRN predicate computation (example)

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

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

reactions:

$$X_1 + X_2 \rightarrow Y$$
$$Y + N \rightarrow Y$$
$$X_1 + Y \rightarrow X_1 + N$$
$$X_2 + Y \rightarrow X_2 + N$$



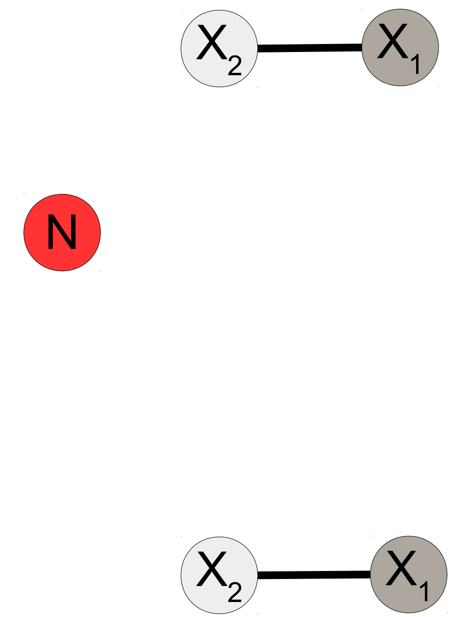
CRN predicate computation (example)

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

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

reactions:

$$\boxed{X_1 + X_2 \rightarrow Y}$$
$$Y + N \rightarrow Y$$
$$X_1 + Y \rightarrow X_1 + N$$
$$X_2 + Y \rightarrow X_2 + N$$



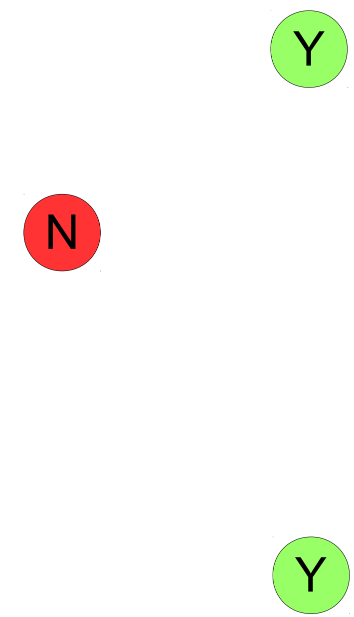
CRN predicate computation (example)

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

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

reactions:

$$\boxed{X_1 + X_2 \rightarrow Y}$$
$$Y + N \rightarrow Y$$
$$X_1 + Y \rightarrow X_1 + N$$
$$X_2 + Y \rightarrow X_2 + N$$



CRN predicate computation (example)

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

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

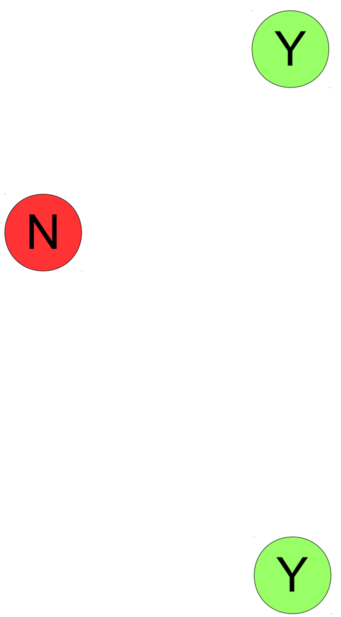
reactions:

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

$$Y + N \rightarrow Y$$

$$\cancel{X_1 + Y \rightarrow X_1 + N}$$

$$\cancel{X_2 + Y \rightarrow X_2 + N}$$



CRN predicate computation (example)

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

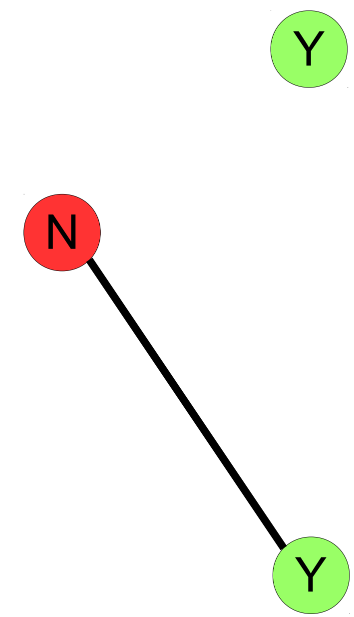
initial state: $\{ x_1\ X_1\ ,\ x_2\ X_2\ ,\ 1\ Y \}$

reactions: $X_1 + X_2 \rightarrow Y$

$\boxed{Y + N \rightarrow Y}$

$\cancel{X_1 + Y \rightarrow X_1 + N}$

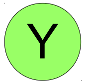
$\cancel{X_2 + Y \rightarrow X_2 + N}$



CRN predicate computation (example)

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

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

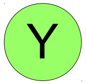


reactions: $X_1 + X_2 \rightarrow Y$

$\boxed{Y + N \rightarrow Y}$

$\cancel{X_1 + Y \rightarrow X_1 + N}$

$\cancel{X_2 + Y \rightarrow X_2 + N}$



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: $p(x_1,x_2)$: "$\boxed{3x_1} > x_2/2$"?

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

reactions: $\boxed{X_1 \to 3Z_1}$

CRN predicate computation (example)

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

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

reactions: $X_1 \rightarrow 3Z_1$

$\boxed{2X_2 \rightarrow Z_2}$

CRN predicate computation (example)

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

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

reactions:

$$X_1 \to 3Z_1$$

$$2X_2 \to Z_2$$

$$\boxed{\begin{array}{l} N + Z_1 \to Y \\ Y + Z_2 \to N \end{array}}$$

Stable predicate computation (definition)

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

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

Stable predicate computation (definition)

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

- **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

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

- **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 φ(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 $p(x_1,\ldots,x_k) \in \{\text{yes},\text{no}\}$, $\quad x_1,\ldots,x_k \in \mathbb{N}$

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 φ(s) of state s: consensus vote (if voters unanimous)

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

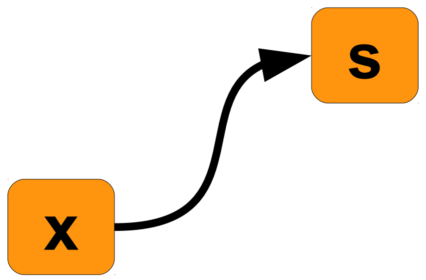
stable computation: for all states **s** reachable from the initial state **x**, a correct output-stable state **o** is reachable from **s**

x

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

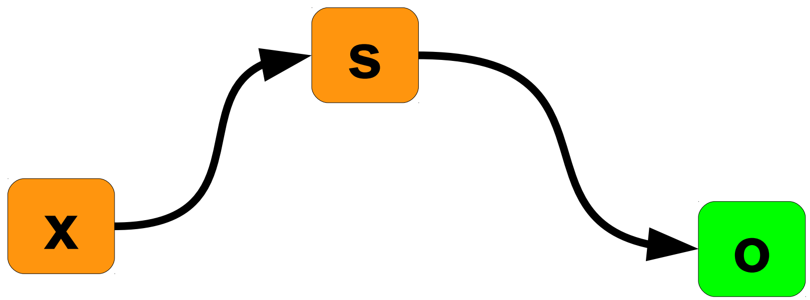
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

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

output φ(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**



$\varphi(\boldsymbol{o}) = p(x_1,\ldots,x_k)$

Stable predicate computation (definition)

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

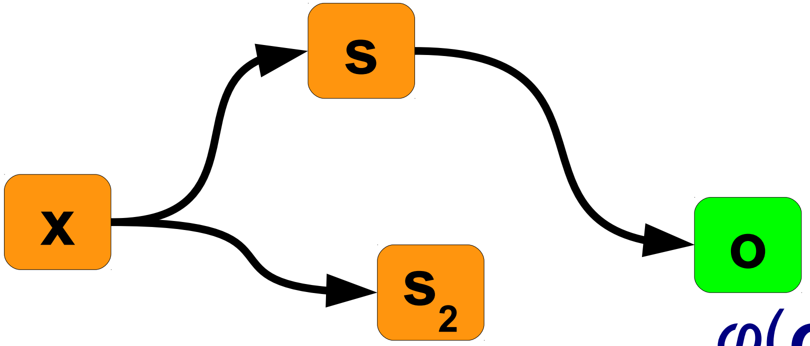
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

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

output φ(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**



$\varphi(\boldsymbol{o}) = p(x_1,..,x_k)$

Stable predicate computation (definition)

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

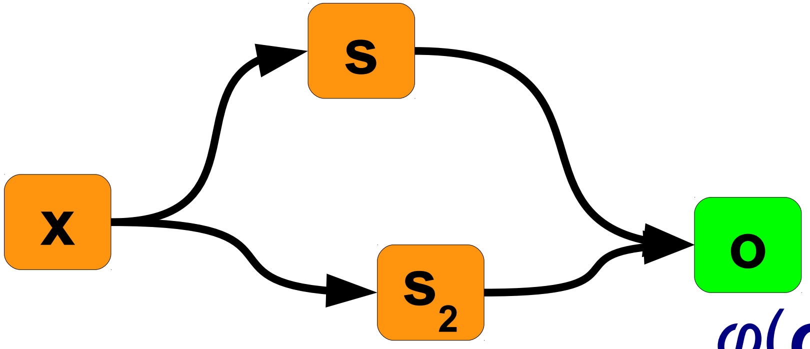
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

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

output φ(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**



$\varphi(\boldsymbol{o}) = p(x_1,\ldots,x_k)$

Stable predicate computation (definition)

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

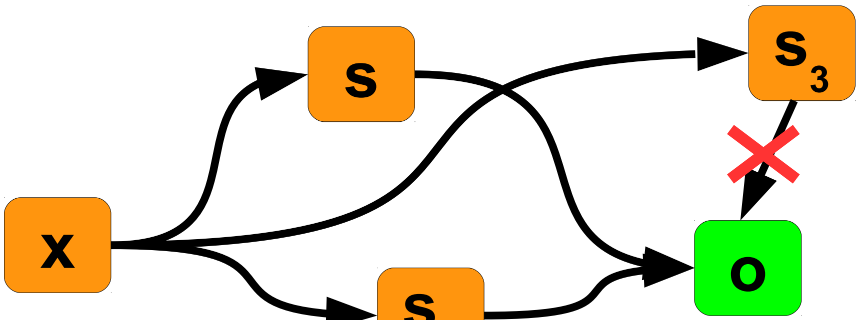
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

votes: two disjoint subsets of species: "yes" and "no" voters

output φ(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**



$\varphi(\boldsymbol{o}) = p(x_1,\ldots,x_k)$

Stable predicate computation (definition)

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

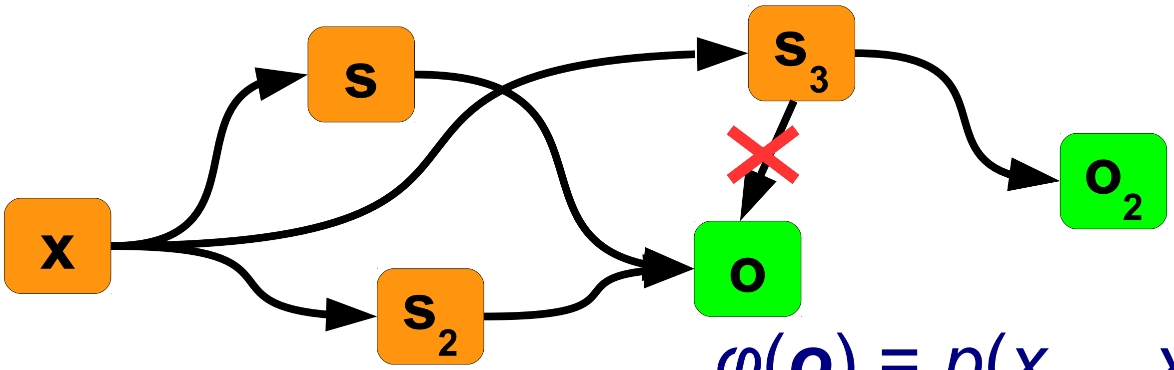
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

votes: two disjoint subsets of species: "yes" and "no" voters

output φ(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**



$\varphi(\mathbf{o}) = p(x_1,\ldots,x_k) = \varphi(\mathbf{o}_2)$

Stable ~~predicate~~ computation (definition)
function

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

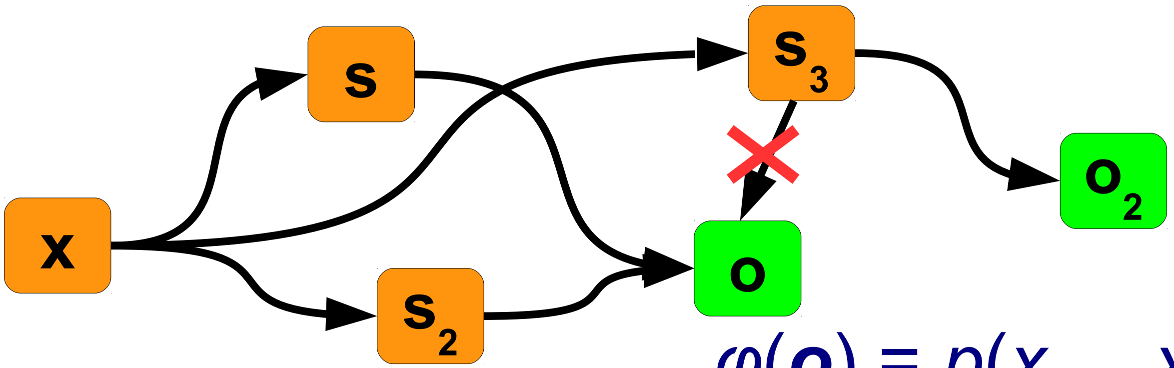
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

votes: two disjoint subsets of species: "yes" and "no" voters

output φ(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**



$\varphi(\mathbf{o}) = p(x_1,\ldots,x_k) = \varphi(\mathbf{o}_2)$

Stable ~~predicate~~ computation (definition)

function

$\mathbb{N}$

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

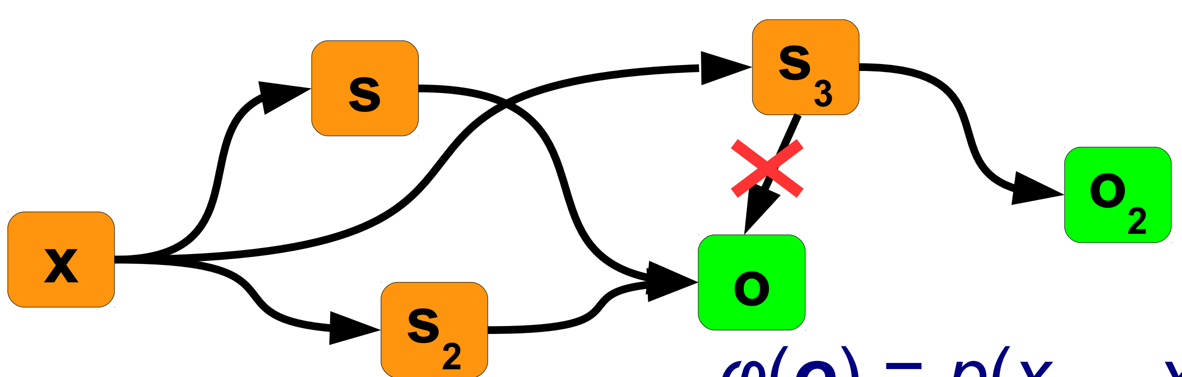
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

votes: two disjoint subsets of species: "yes" and "no" voters

output φ(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**



$\varphi(\mathbf{o}) = p(x_1,\ldots,x_k) = \varphi(\mathbf{o}_2)$

Stable ~~predicate~~ computation (definition)

function $\mathbb{N}$

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

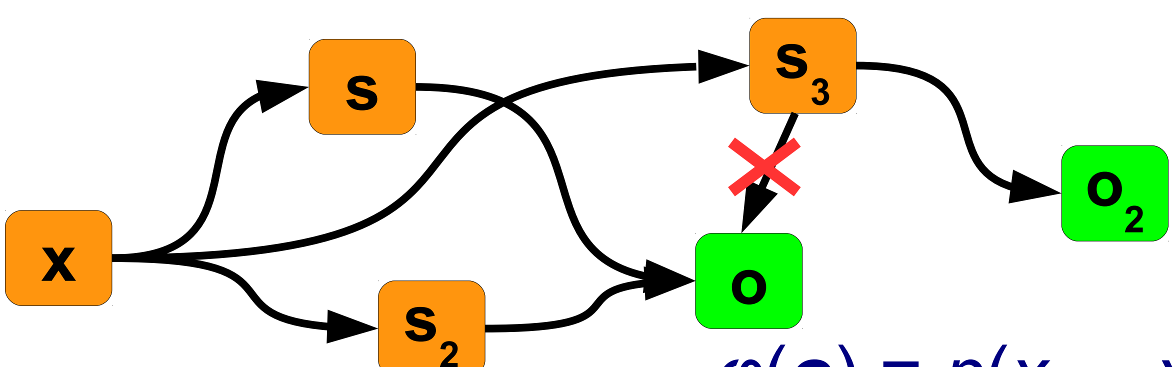
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

~~**votes**: two disjoint subsets of species: "yes" and "no" voters~~

output φ(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**



$\varphi(\mathbf{o}) = p(x_1,\ldots,x_k) = \varphi(\mathbf{o}_2)$

Stable ~~predicate~~ function computation (definition)

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

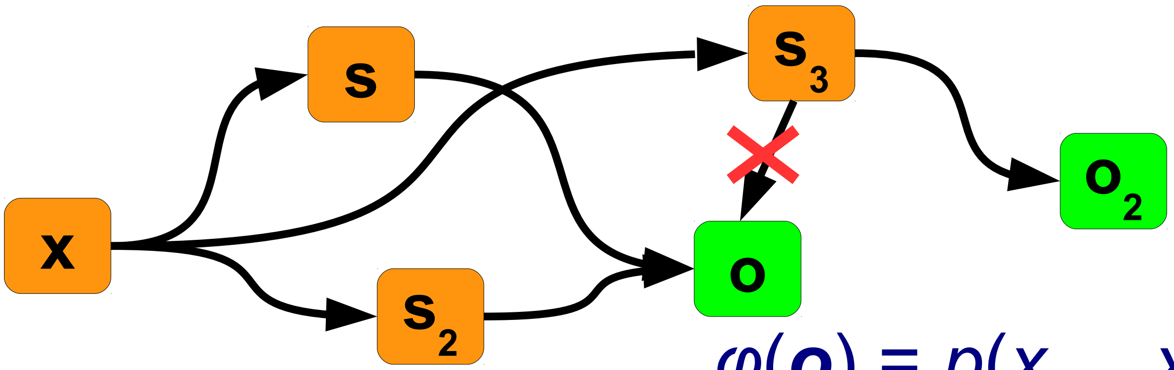
initial state: $\#X_1 = x_1,\ldots,\#X_k = x_k$, constant counts of other species

~~**votes**: two disjoint subsets of species: "yes" and "no" voters~~

output φ(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**



$\varphi(\mathbf{o}) = p(x_1,\ldots,x_k) = \varphi(\mathbf{o}_2)$

Stable computation characterization

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

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

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

[Chen, D, Soloveichik, DNA Computing 2012]

predicates

functions

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

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

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

[Chen, D, Soloveichik, DNA Computing 2012]

predicates

functions

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

$x_1 - 3x_2 < -7$

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

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

predicates

[Chen, D, Soloveichik, DNA Computing 2012]

functions

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

$2x_1 + x_2 \equiv 3 \bmod 5$ $\quad$ $x_1 - 3x_2 < -7$

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

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

[Chen, D, Soloveichik, DNA Computing 2012]

predicates

functions

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

$2x_1 + x_2 \equiv 3 \bmod 5$ $\quad$ $x_1 - 3x_2 < -7$

semilinear function ≈ piecewise linear

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

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

predicates

[Chen, D, Soloveichik, DNA Computing 2012]

functions

Real-valued CRNs

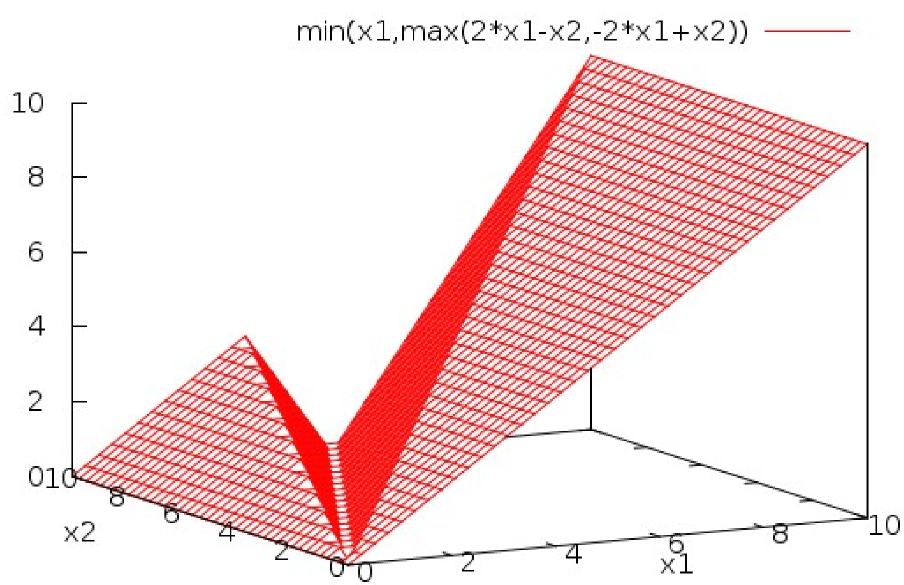
Theorem from previous slide: A function is stably computed by a integer-valued CRN if and only if it is *semilinear*.

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]



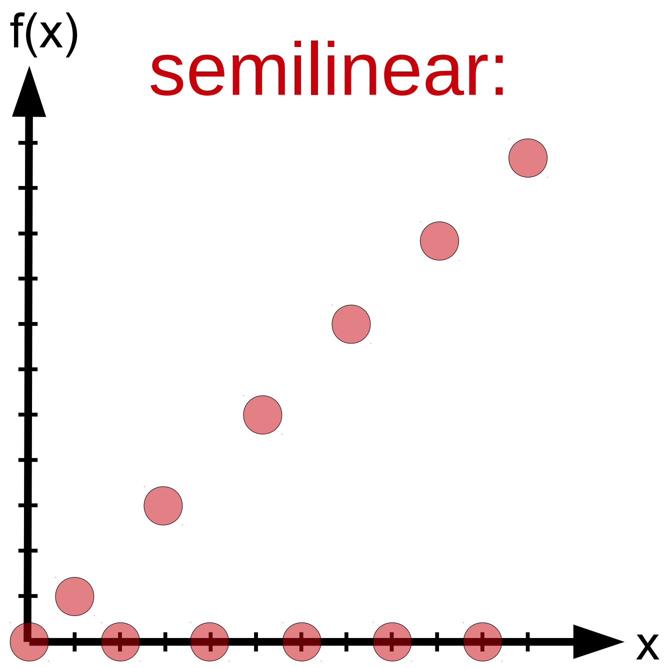
Real-valued CRNs

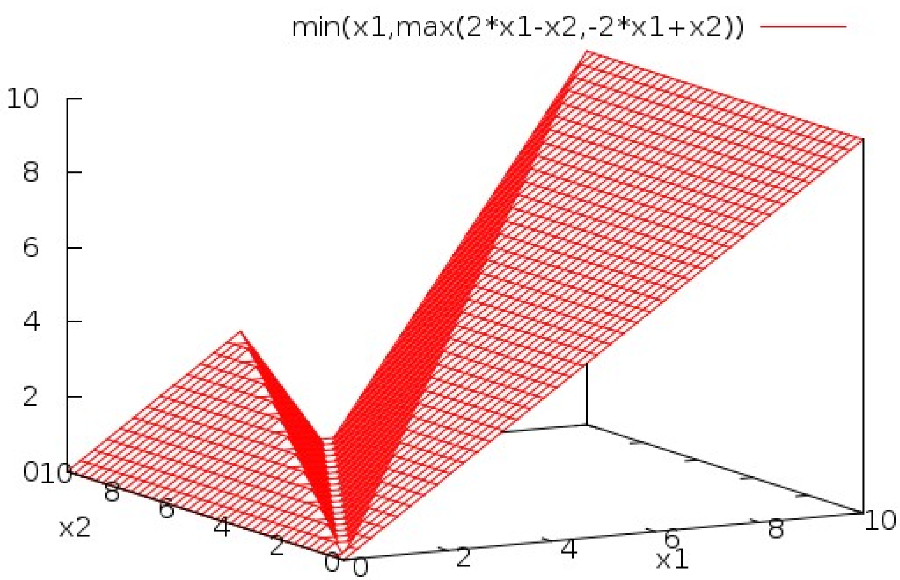
Theorem from previous slide: A function is stably computed by a integer-valued CRN if and only if it is *semilinear*.

≈ 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]





Time complexity of stable computation

n = # molecules in initial state ≈ volume

Time complexity of stable computation

n = # molecules in initial state ≈ 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 = # molecules in initial state ≈ 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 = # molecules in initial state ≈ 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

<u>reaction</u>	<u>expected time</u>
$X \rightarrow W + 2Y + Z$	$1 / \#X$
$A + B \rightarrow X$	volume / $(\#A \bullet \#B)$

Time complexity (example)

$$\{n\ X\}$$
$$X \rightarrow Y + Y$$

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[time to consume all *X*] = E[time to consume 1st X]
+ E[time to consume 2nd X]
+ E[time to consume 3rd X]
+ ...
+ E[time to consume final X]

Time complexity (example)

$$\{n\ X\}$$
$$X \to Y + Y$$

E[time to consume all X] =	E[time to consume 1st X]	1/n
	+ E[time to consume 2nd X]	1/(n-1)
	+ E[time to consume 3rd X]	1/(n-2)
	+ ...	
	+ E[time to consume final X]	1/1

Time complexity (example)

$$\{n\ X\}$$
$$X \rightarrow Y + Y$$

E[time to consume all X] = E[time to consume 1st X]   1/n
+ E[time to consume 2nd X]   1/(n-1)
+ E[time to consume 3rd X]   1/(n-2)
+ ...
+ E[time to consume final X]   1/1

≈ log n

Time complexity (example)

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

$X + X \to Y$

Time complexity (example)

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

$X + X \rightarrow Y$

"finite density constraint"

Time complexity (example)

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

$X + X \rightarrow Y$

"finite density constraint"

volume

$\#X^2$

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

Time complexity (example)

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

$X + X \rightarrow Y$

"finite density constraint"

volume

#X²

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

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

Time complexity (example)

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

$X + X \rightarrow Y$

"finite density constraint"

volume $\#X^2$

$$
\begin{aligned}
\text{E[time to consume all } X\text{]} &= n/n^2 + n/(n-2)^2 + n/(n-4)^2 + \ldots + n \\
&< n(1/2^2 + 1/4^2 + 1/6^2 + 1/8^2 + \ldots) \\
&= O(n)
\end{aligned}
$$

Time complexity (example)

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

$X + X \to Y$

"finite density constraint"

volume

#X²

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

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

$= O(n) = \Theta(n)$

Time complexity (example)

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

$X + X \to Y$

"finite density constraint"

$$\text{E[time to consume all } X] = \underbrace{n}_{\text{volume}}/\underbrace{n^2}_{\#X^2} + n/(n-2)^2 + n/(n-4)^2 + \ldots + n$$

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

$$= O(n) = \Theta(n)$$

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

(Open problem)

Time complexity (leader election)

$\{n\ L\}$, volume n

$L + L \to L$

Time complexity (leader election)

$\{n\ L\}$, volume n

$L + L \rightarrow L$

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

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?

Time complexity (leader election)

$\{n\ L\}$, volume n

$L + L \to L$

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

Is there a faster CRN?

- If we really abuse the CRN model, probably (use $2X \to 3X$)

Time complexity (leader election)

$\{n\ L\}$, volume n

$L + L \to L$

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

Is there a faster CRN?

- If we really abuse the CRN model, probably (use $2X \to 3X$)
- In mass-conserving CRNs, we don't know

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 $2X \rightarrow 3X$)
- 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)

$\{n\ L\}$, volume n
$L + L \to L$

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

Is there a faster CRN?

- If we really abuse the CRN model, probably (use $2X \to 3X$)
- 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

$L + L \to L$

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

Is there a faster CRN?

- If we really abuse the CRN model, probably (use $2X \to 3X$)
- In mass-conserving CRNs, we don't know
  - [*Angluin, Aspnes, Eisenstat, <u>Symposium on Distributed Computing</u> 2006*] show a CRN that seems to work in simulation, with small probability of error (*unproven*)
  - [*Alistarh and Gelashvili, <u>International Colloquium on Automata, Languages, and Programming</u> 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, <u>in submission</u>*]

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

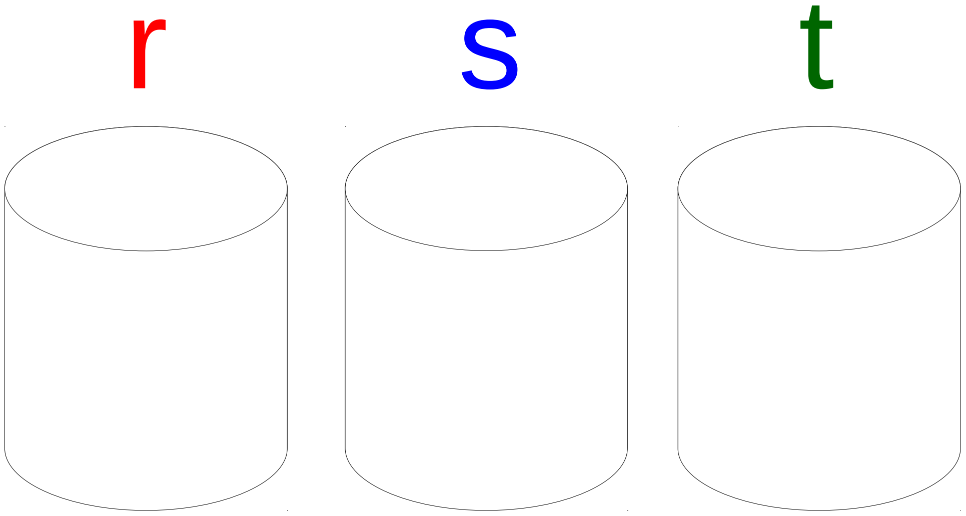
(Informally) A CRN can simulate any algorithm, with a small chance of error.

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

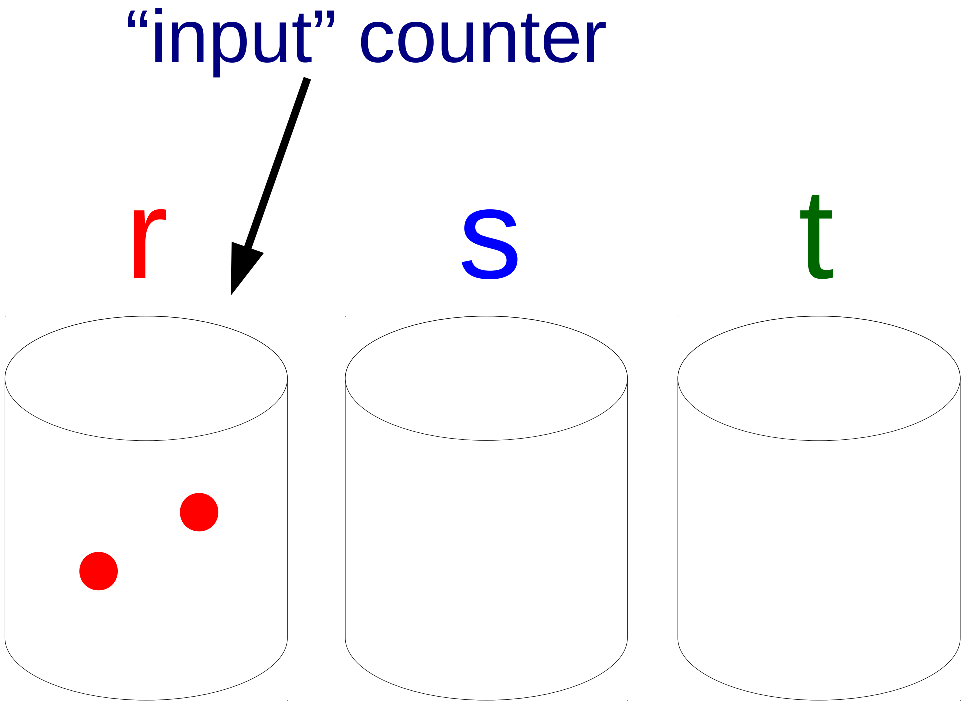
[Angluin, Aspnes, Eisenstat, Symposium on Distributed Computing 2006]
[Soloveichik, Cook, Winfree, Bruck, Natural Computing 2008]

Counter (register) machine

Counter (register) machine



Counter (register) machine



Counter (register) machine

1) *dec*(r)

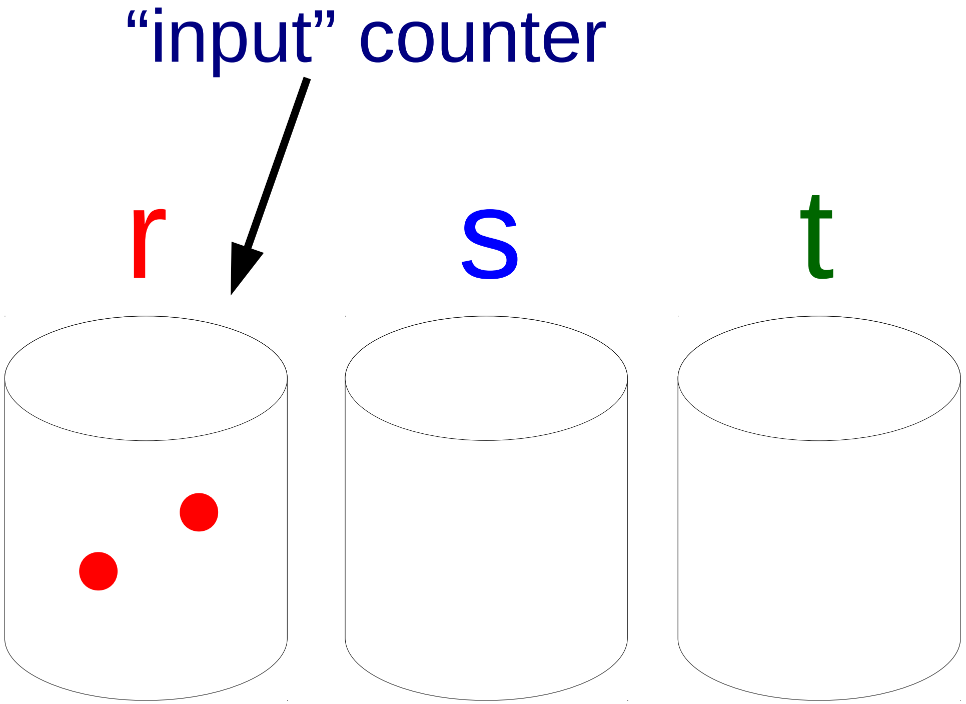
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

5) *dec*(t)

6) *inc*(s)



Counter (register) machine

1) *dec*(r)

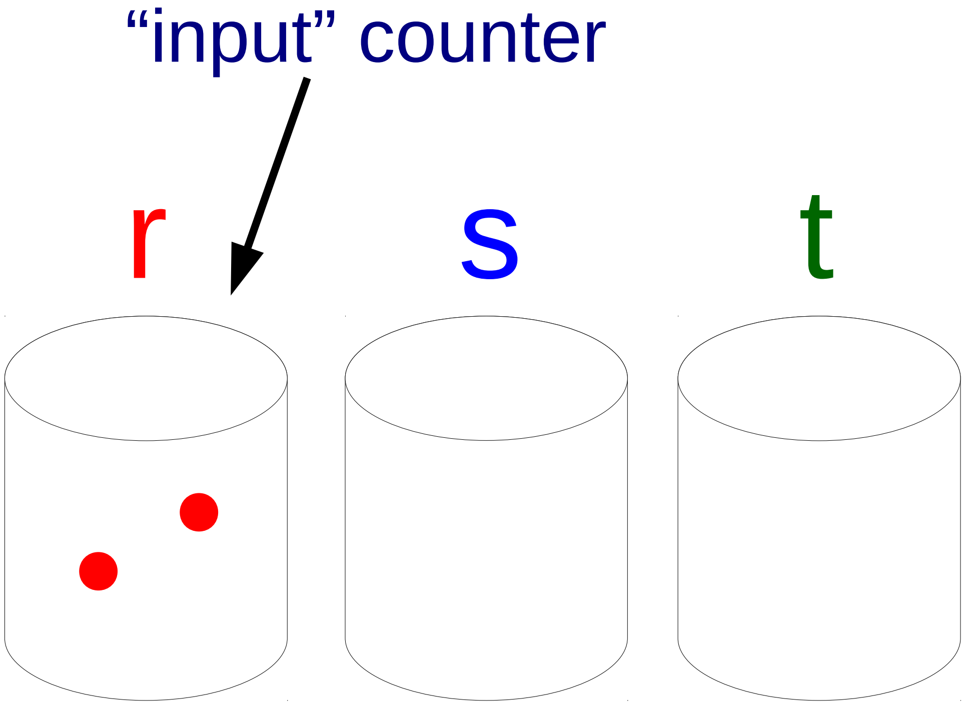
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

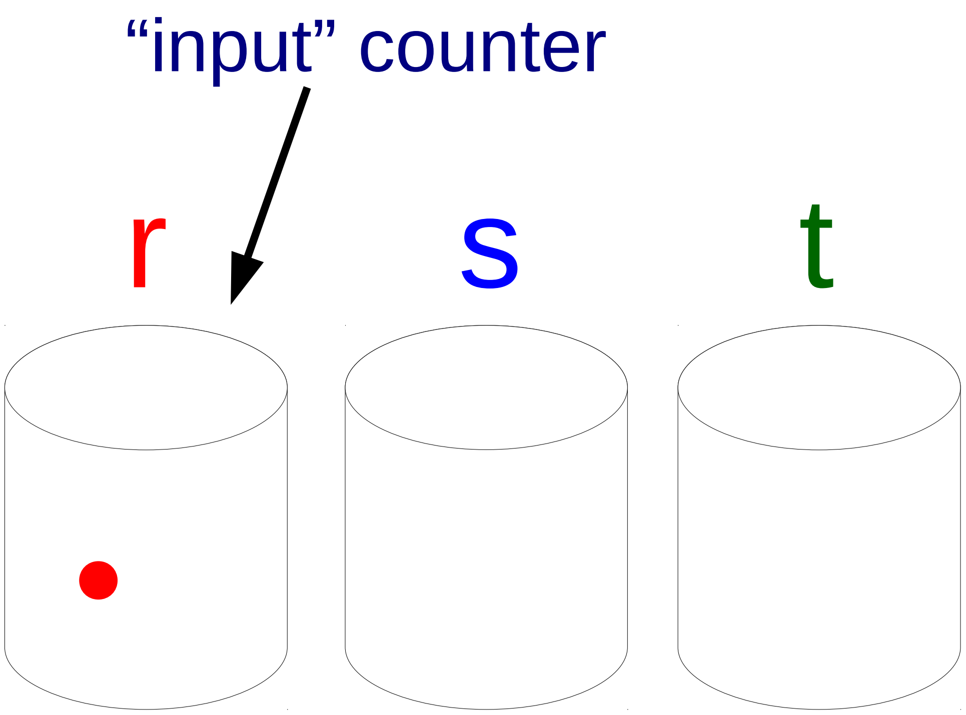
5) *dec*(t)

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)

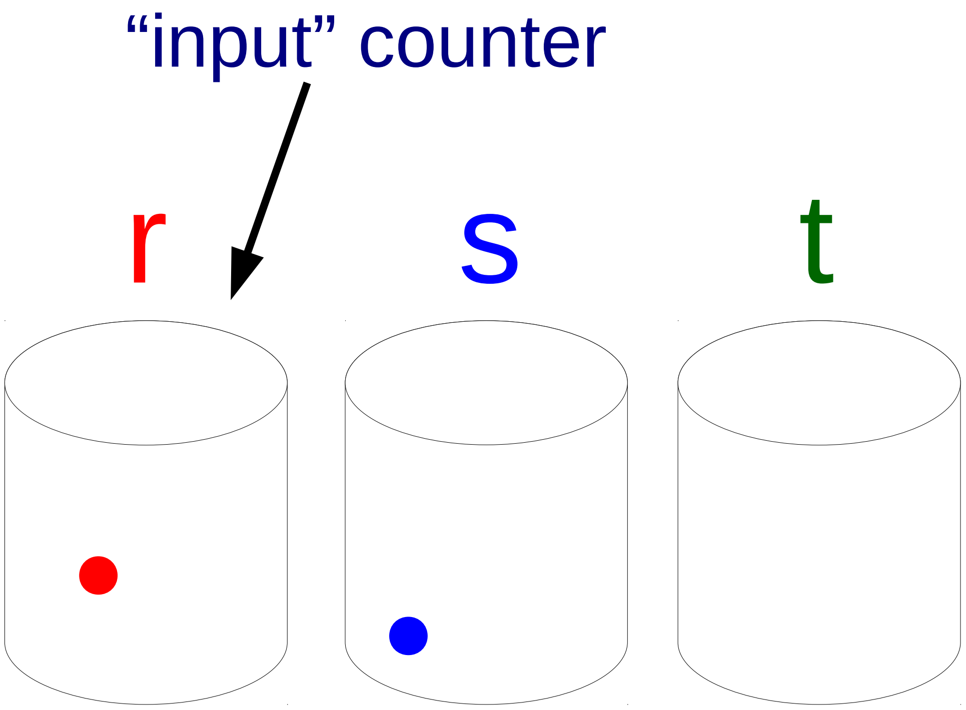
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

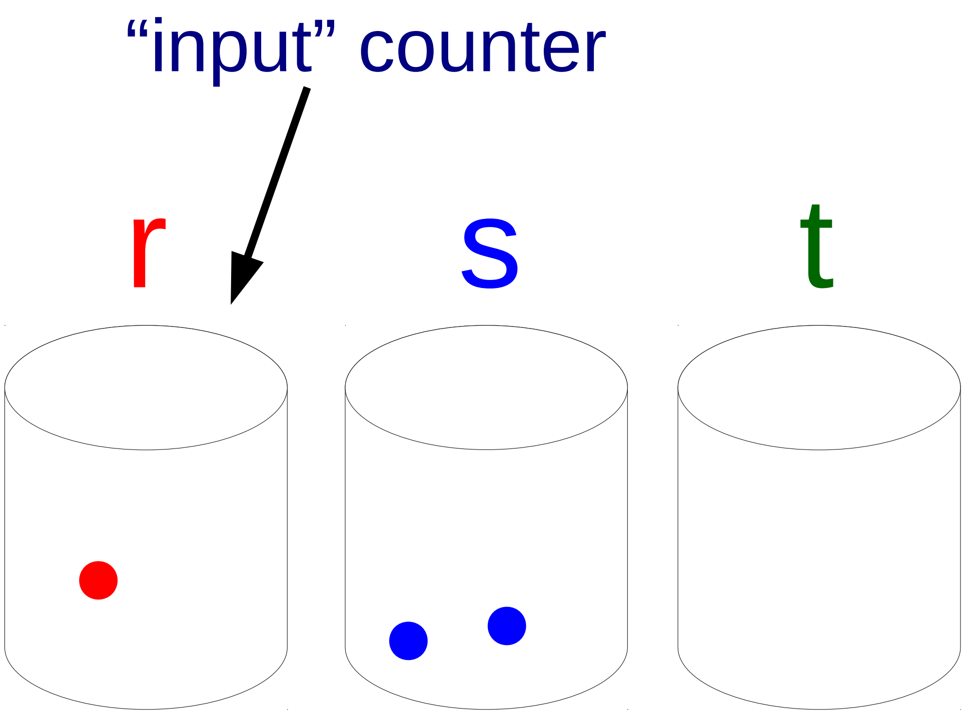
5) *dec*(t)

6) *inc*(s)



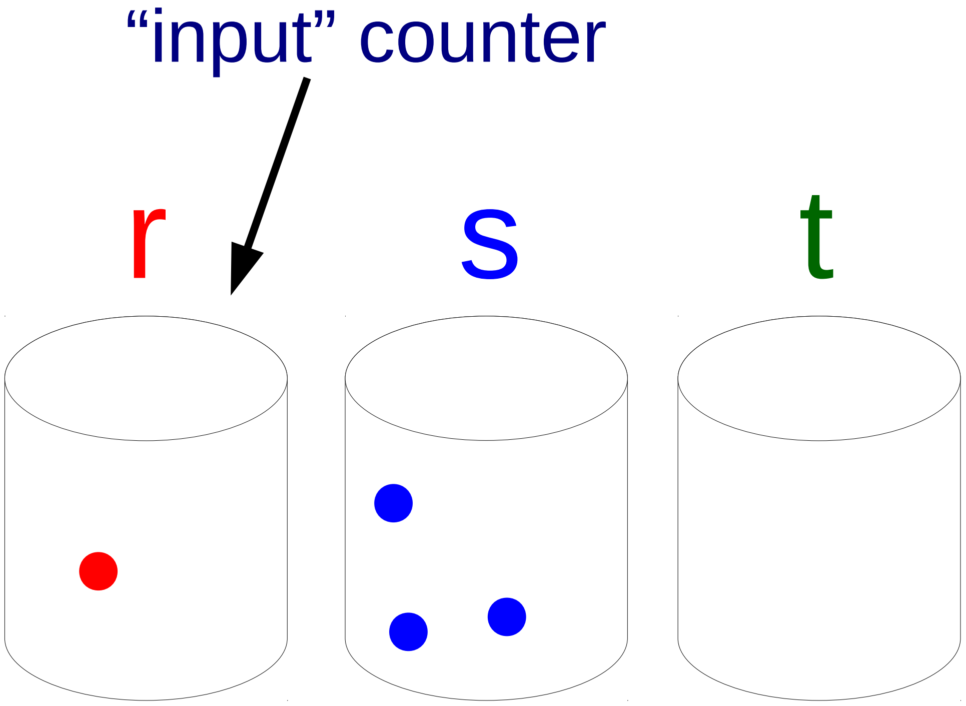
Counter (register) machine

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)

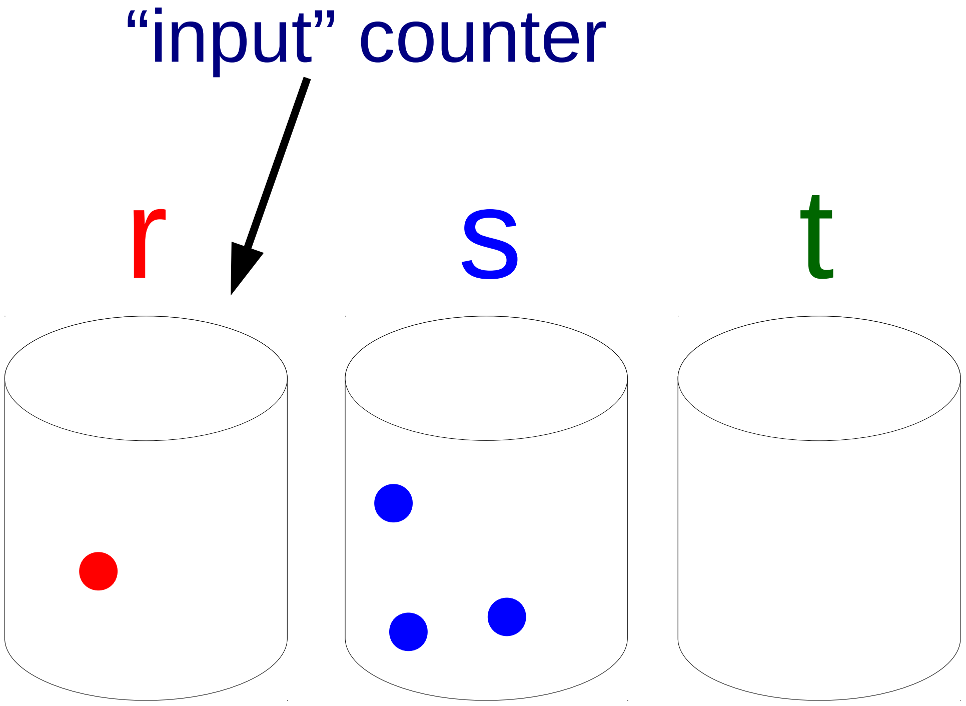
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

5) *dec*(t)

6) *inc*(s)



Counter (register) machine

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

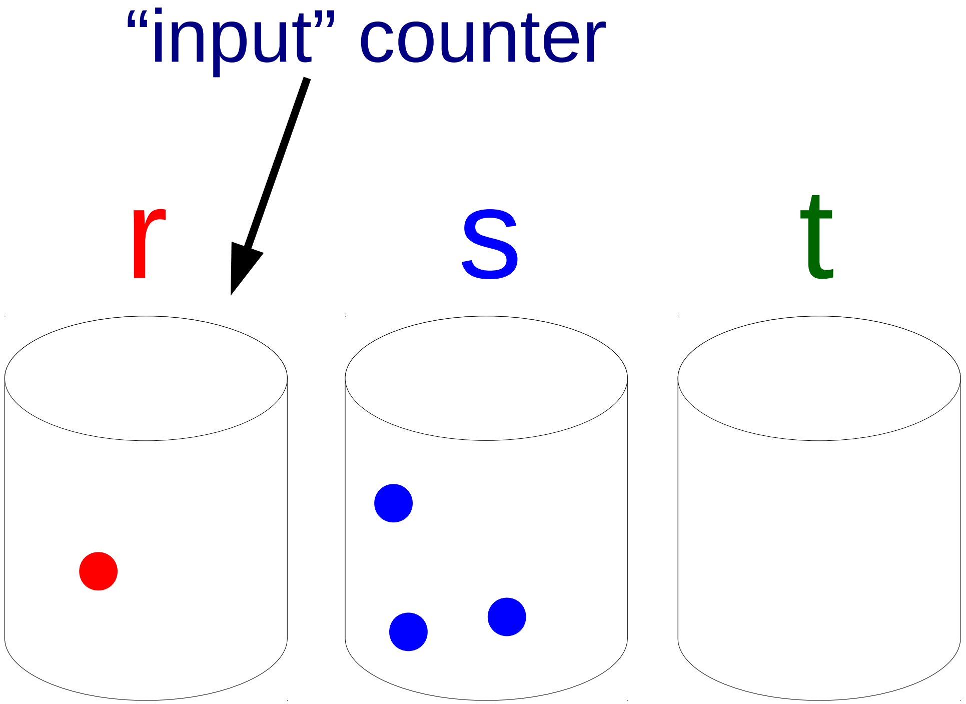
2) *inc*(s)

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

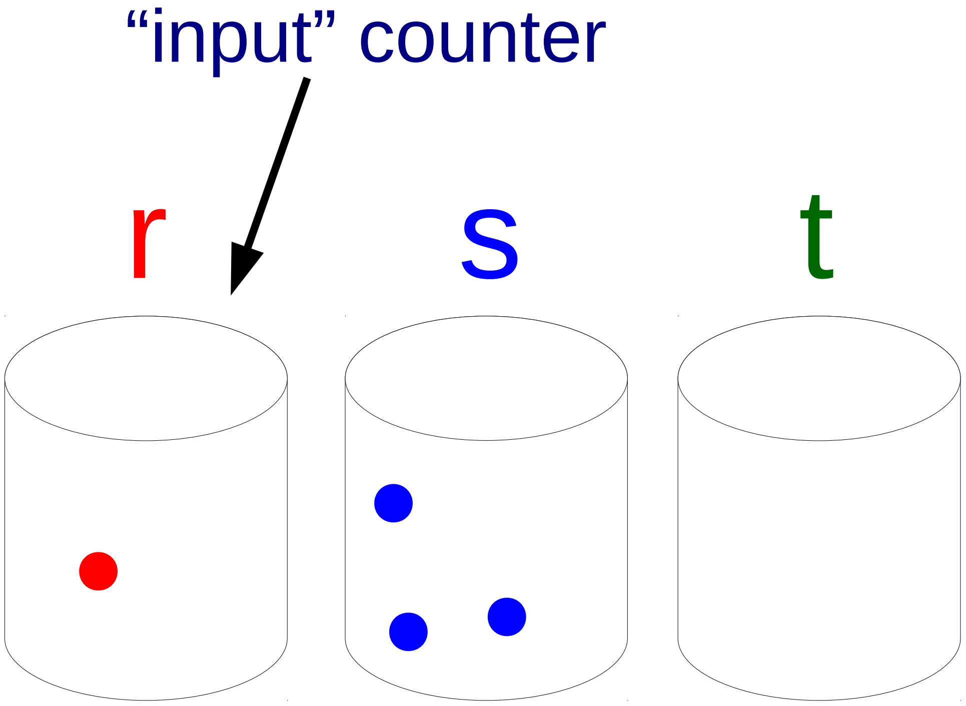
2) *inc*(s)

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

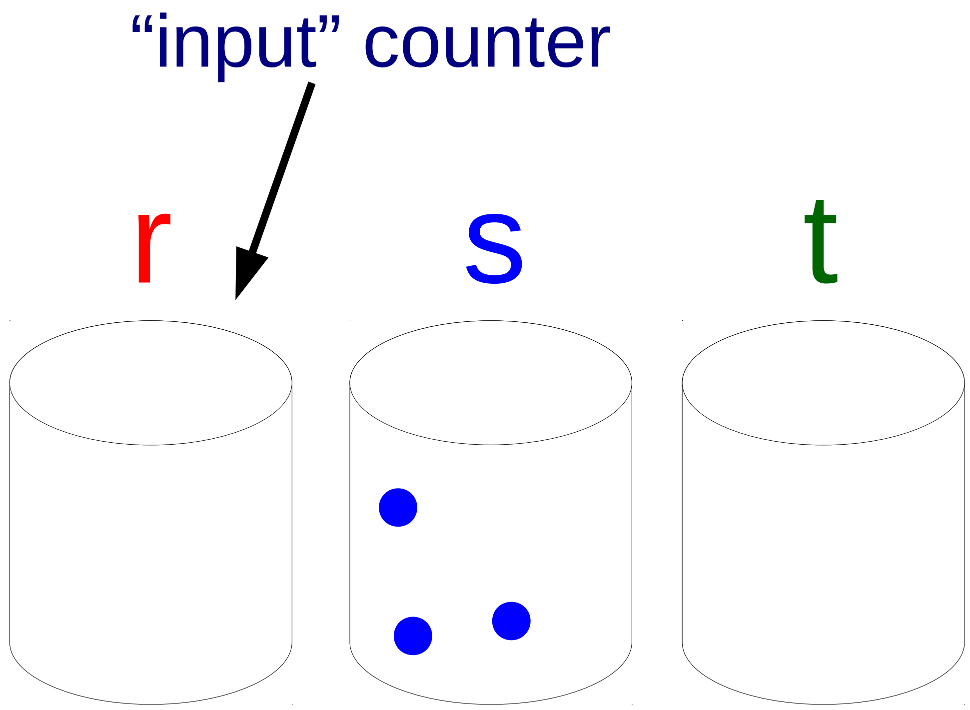
2) *inc*(s)

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

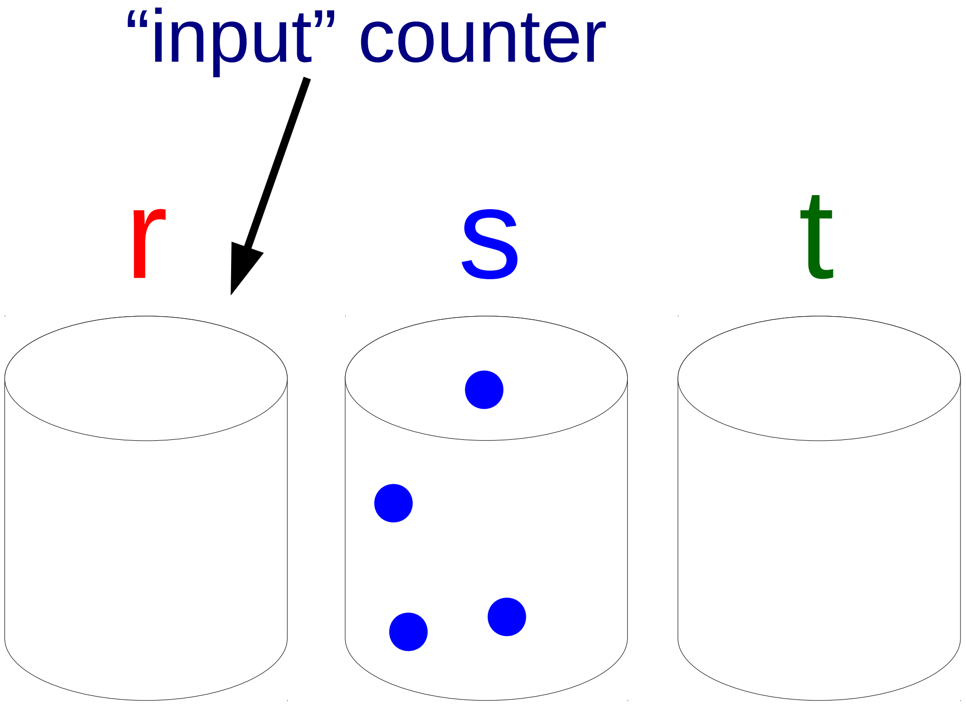
2) *inc*(s)

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

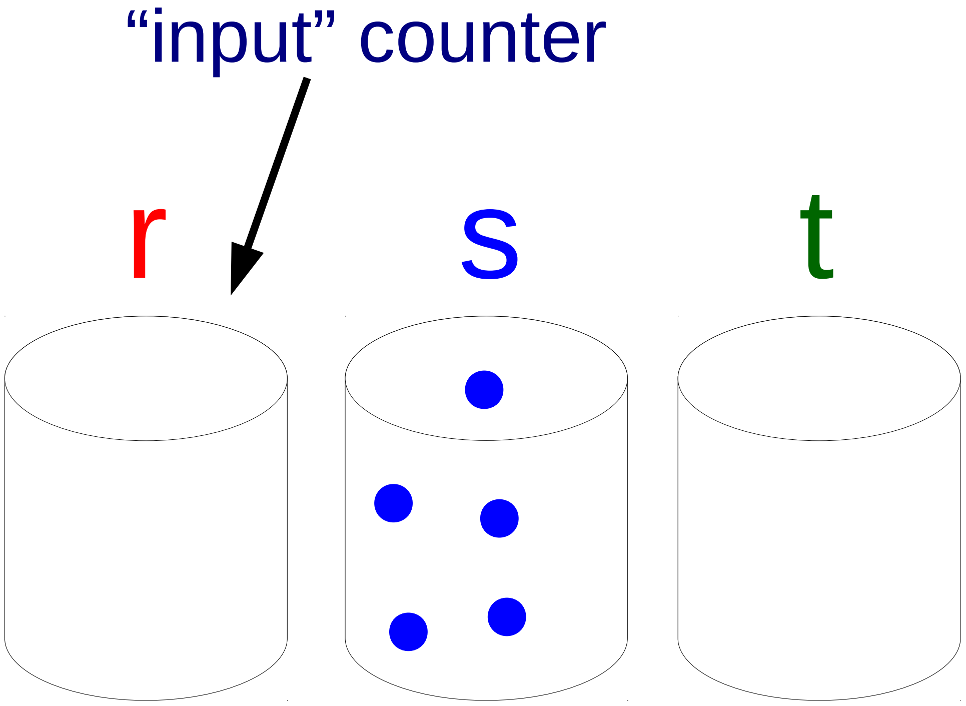
2) *inc*(s)

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

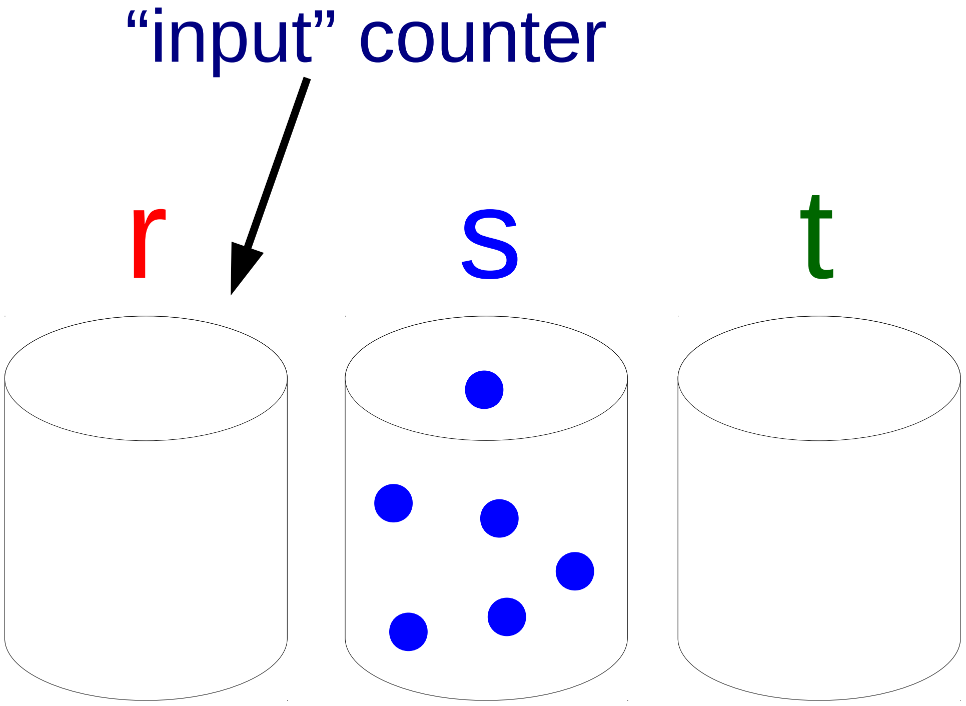
2) *inc*(s)

3) *inc*(s)

4) *inc*(s)

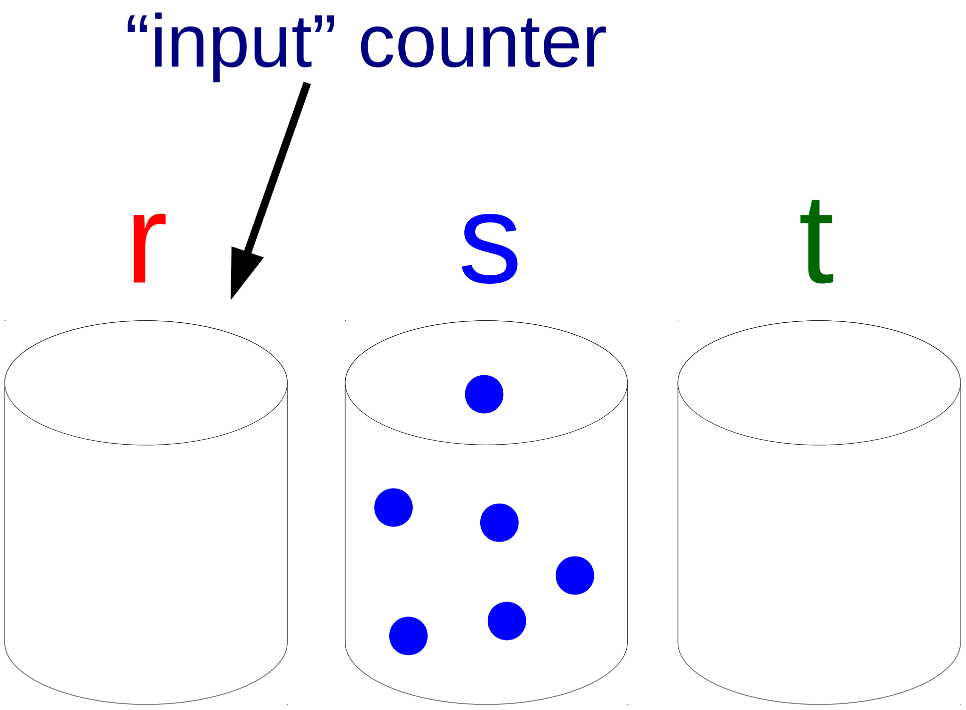
5) *dec*(t) if empty goto 1

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

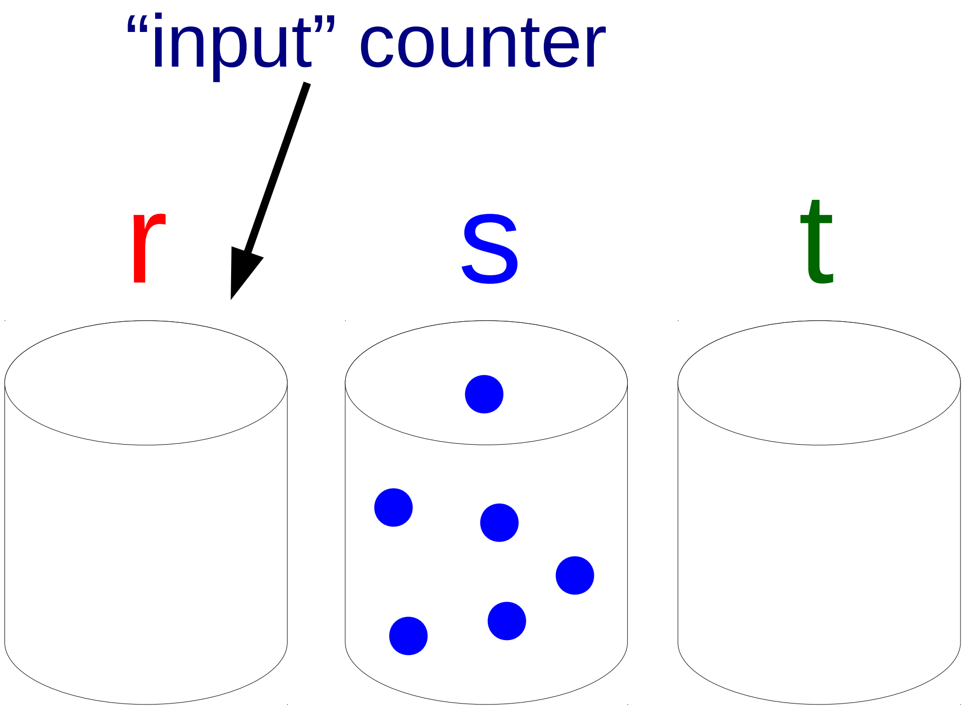
2) *inc*(s)

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

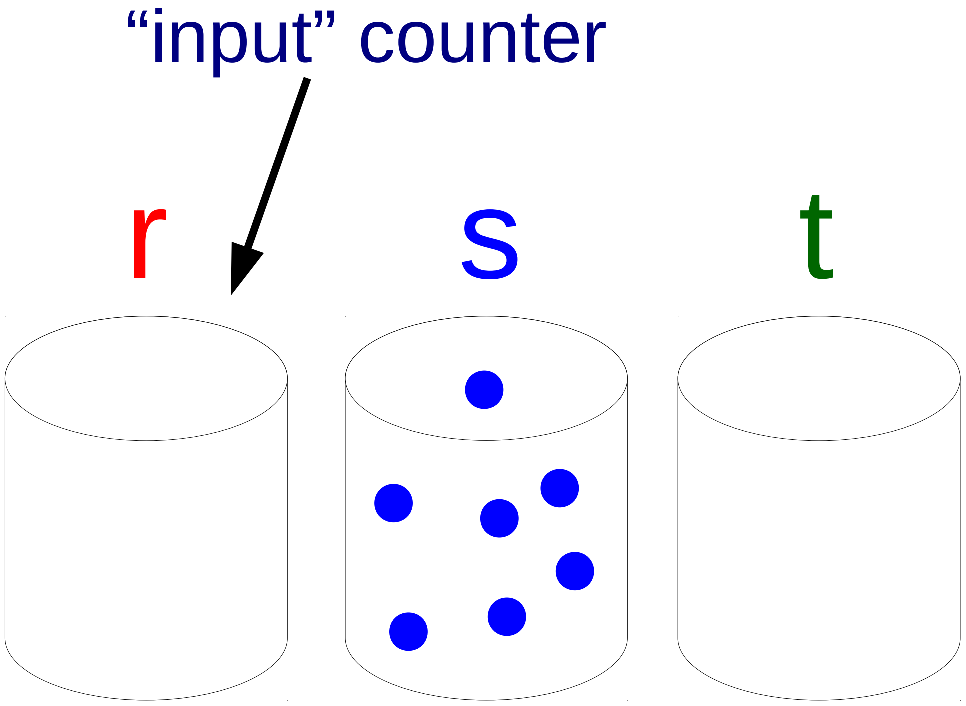
2) *inc*(s)

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)

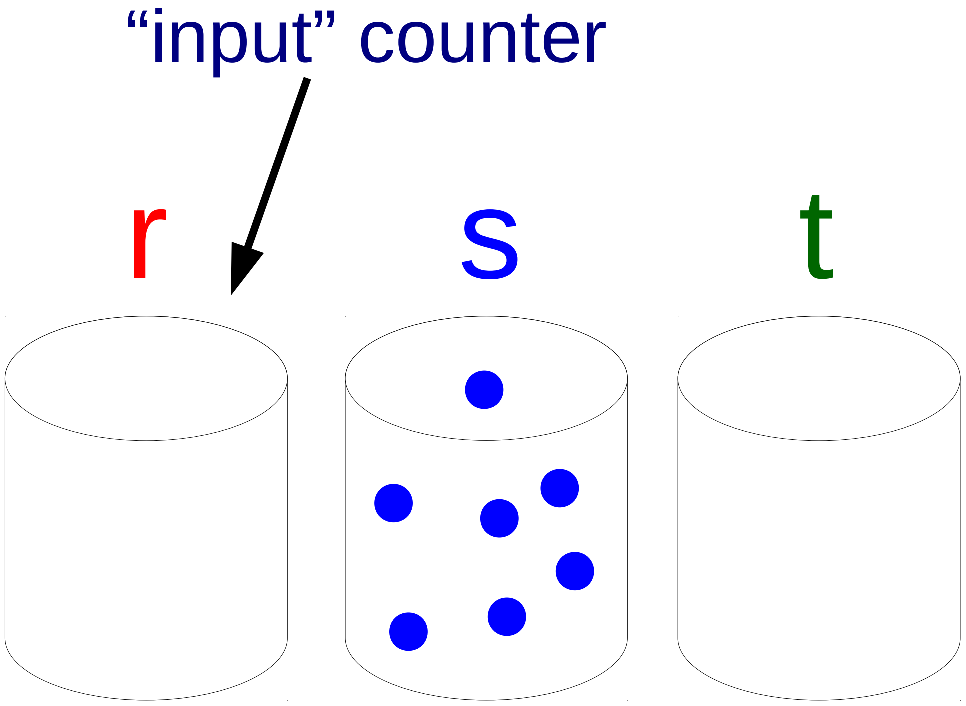
3) *inc*(s)

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)

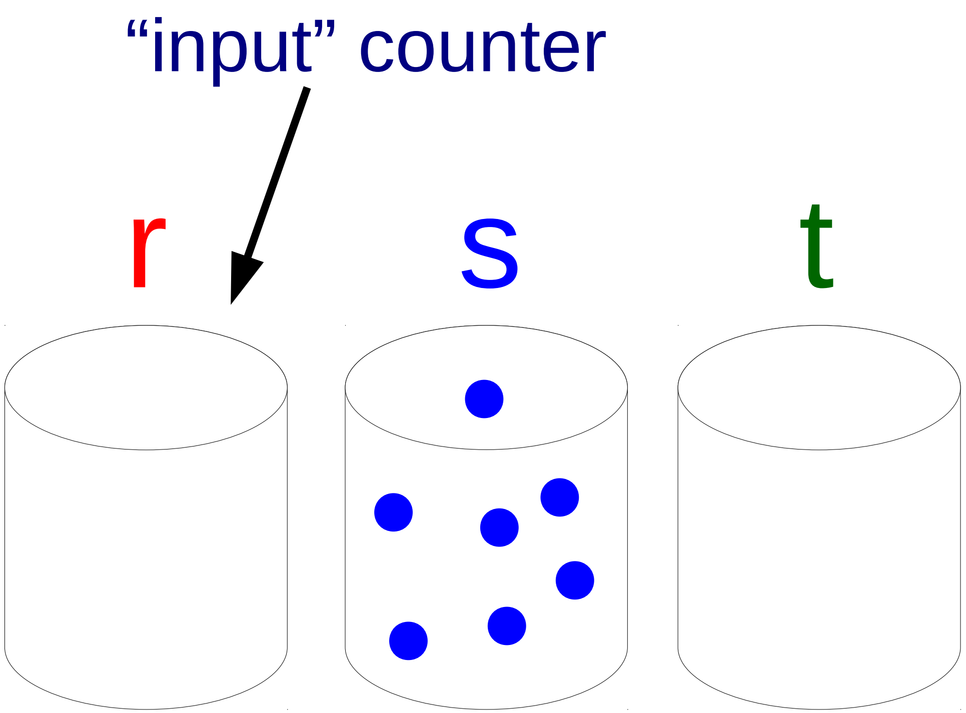
3) *inc*(s)

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

$L_1 \to L_2 + R$

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

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

$L_1 \rightarrow L_2 + R$

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

$L_3 \rightarrow L_4 + S$

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

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

$L_1 \to L_2 + R$

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

$L_3 \to L_4 + S$

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

Need to be **very** slow!

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

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

Use a clock:

1 C_1, 1 F

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

Use a clock:

1 C_1, 1 F

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

$$F + C_2 \to F + C_3$$

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

Use a clock:

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

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

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

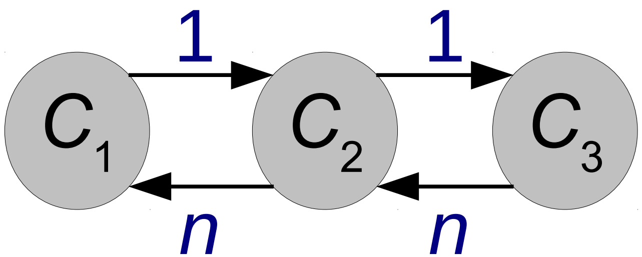
How to slow down reaction $L_2 \rightarrow L_1$?

Use a clock:

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

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

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



reverse-biased random walk

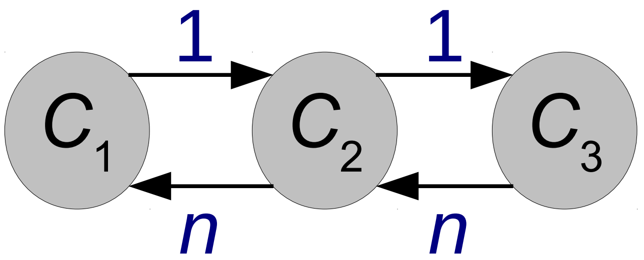
How to slow down reaction $L_2 \to L_1$?

Use a clock:

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

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

$$F + C_2 \to F + C_3 \qquad B + C_3 \to B + C_2$$



reverse-biased random walk

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

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

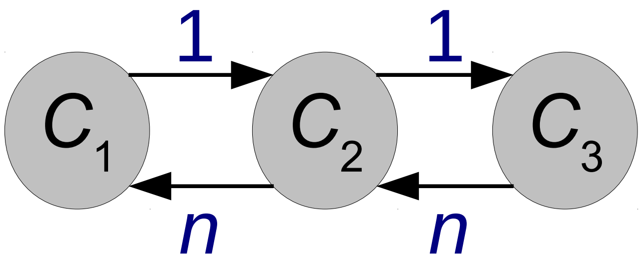
Use a clock:

$$\boxed{C_3 +} L_2 \to \boxed{C_1 +} L_1$$

1 C_1, 1 F , n B

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

$$F + C_2 \to F + C_3 \qquad B + C_3 \to B + C_2$$



reverse-biased random walk

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

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

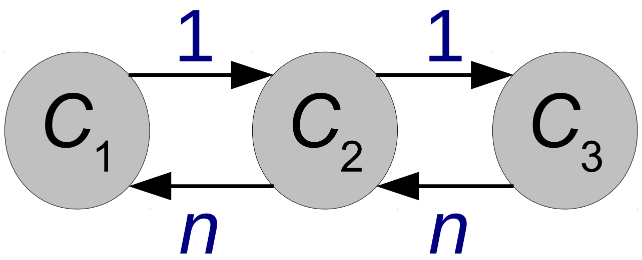
Use a clock:

$\boxed{C_3 + } L_2 \to \boxed{C_1 + } L_1$

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

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

$F + C_2 \to F + C_3$ $\quad$ $B + C_3 \to B + C_2$



reverse-biased random walk

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

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

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...

$$Y \xrightarrow{2} 2Y$$

$$Y \xrightarrow{1}$$

stably computes φ = no (prob < 1)

initial state {1Y,1N}

Probability 1 computation

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

Consider...

$$Y \xrightarrow{2} 2Y$$
$$Y \xrightarrow{1}$$
$$\xrightarrow{1} Y$$

computes φ = yes with prob 1 (not stably)

Probability 1 computation

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

Consider...

$$Y \xrightarrow{2} 2Y$$

$$Y \xrightarrow{1}$$

$$\xrightarrow{1} Y$$

computes φ = yes with prob 1 (not stably)

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

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