The computational power of execution bounded chemical reaction networks

David Doty, Ben Heckmann

May 2024

Seminar on the Mathematics of Reaction Networks

UC DAVIS COMPUTER SCIENCE

Acknowledgments

Ben Heckmann
Undergraduate student
Technische Universität München, UC Davis

Matthias Köppe
Professor
UC Davis

For teaching us about
“*Theorems of the Alternative*”

Chemical reaction networks

Chemical reaction networks

reactant(s) $R \rightarrow P_1 + P_2$ *product(s)*

Chemical reaction networks

reactant(s) $R \rightarrow P_1 + P_2$ *product(s)*

monomers $M_1 + M_2 \rightarrow D$ *dimer*

Chemical reaction networks

reactant(s) $R \to P_1 + P_2$ *product(s)*

monomers $M_1 + M_2 \to D$ *dimer*

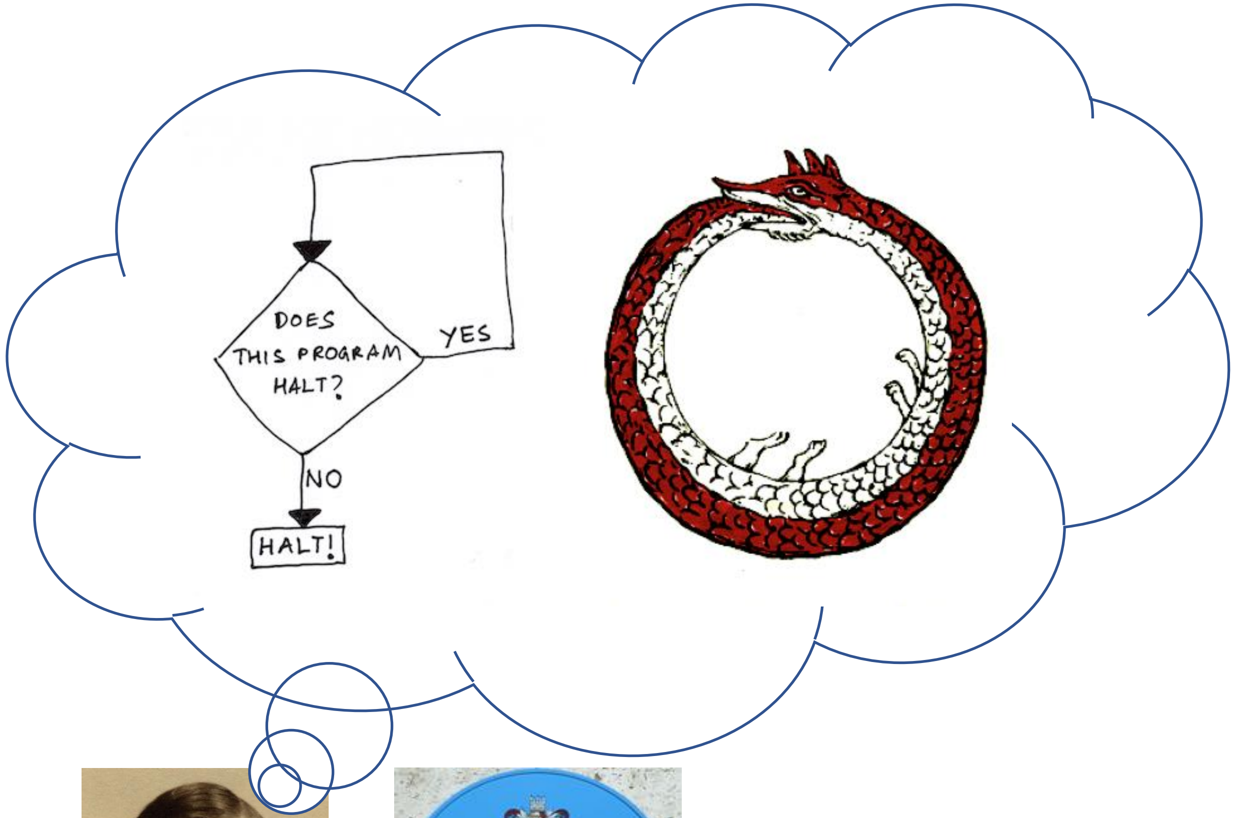
catalyst $C + X \to C + Y$

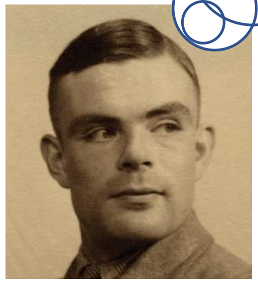
Chemical reaction networks

reactant(s)	$R \rightarrow P_1 + P_2$	*product(s)*
monomers	$M_1 + M_2 \rightarrow D$	*dimer*
catalyst	$C + X \rightarrow C + Y$	

Traditionally a descriptive modeling language…
Let's instead use it as a prescriptive programming language

Theoretical computer science approach





What computation is possible and what is not?

Outline

- <u>**Formal definition of chemical reaction networks**</u>
- Execution bounded chemical reaction networks and linear potential functions
- What is “computation” with chemical reactions?
- Limitations of computation with execution bounded chemical reaction networks

Chemical Reaction Network (CRN)

- finite set of d species $\Lambda = \{ A, B, C, D, \ldots \}$
- finite set of reactions: *e.g.*

$$A+B \to A+C$$
$$C \to A+A$$
$$C+B \to C$$

- state $\mathbf{x} \in \mathbb{N}^d$: molecular counts of each species

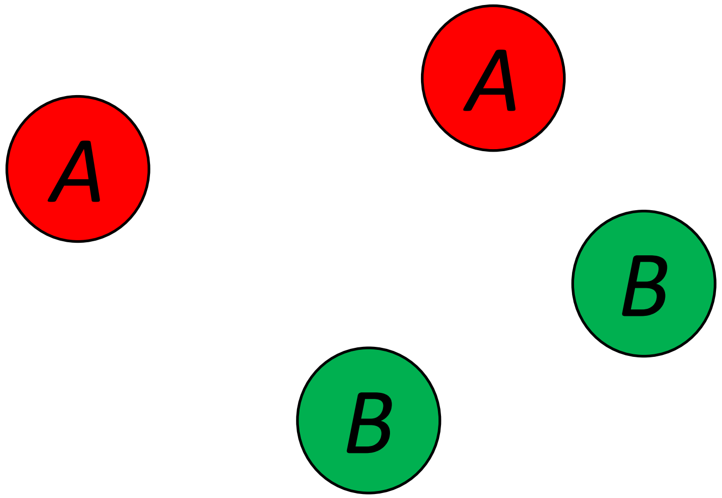
What is possible:
Example execution (reaction sequence)

α: $A + B \rightarrow A + C$

β: $C \rightarrow A + A$

A	B	C

$\mathbf{x} = (2, 2, 0)$

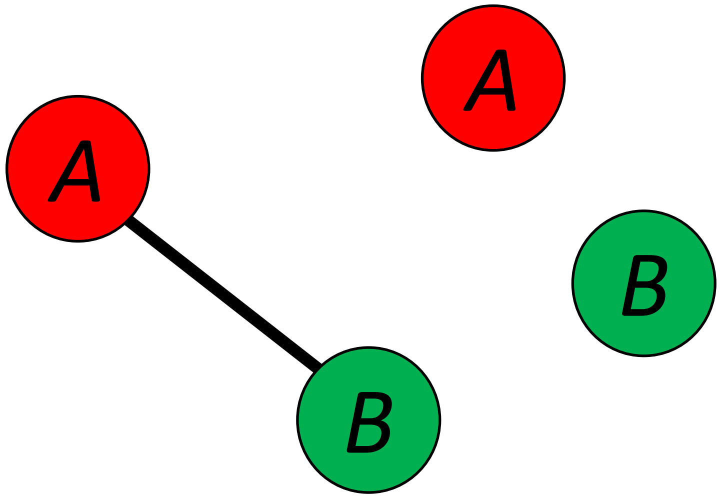


What is possible:
Example execution (reaction sequence)

α: $\boxed{A+B} \rightarrow A+C$

β: $C \rightarrow A+A$

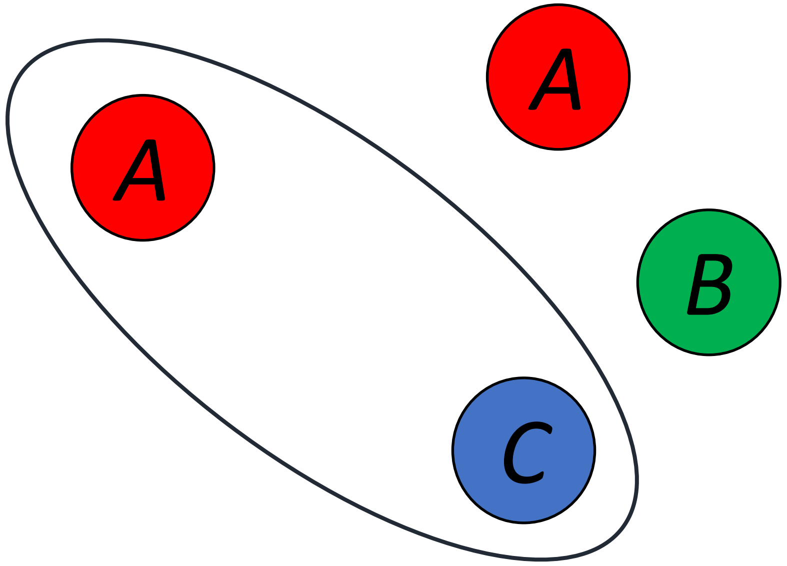
A	B	C
x = (2,	2,	0)



What is possible:
Example execution (reaction sequence)

α: $A+B \rightarrow \boxed{A+C}$

β: $C \rightarrow A+A$

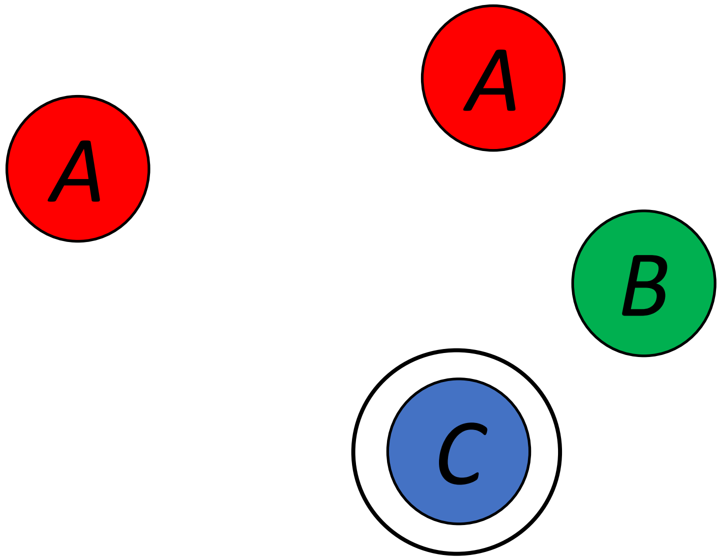


$$\begin{array}{cccc} & A & B & C \\ \mathbf{x} = & (2, & 2, & 0) \\ & \alpha \Downarrow & & \\ & (2, & 1, & 1) \end{array}$$

What is **possible**:
Example execution (reaction sequence)

α: $A+B \to A+C$

β: $\boxed{C} \to A+A$



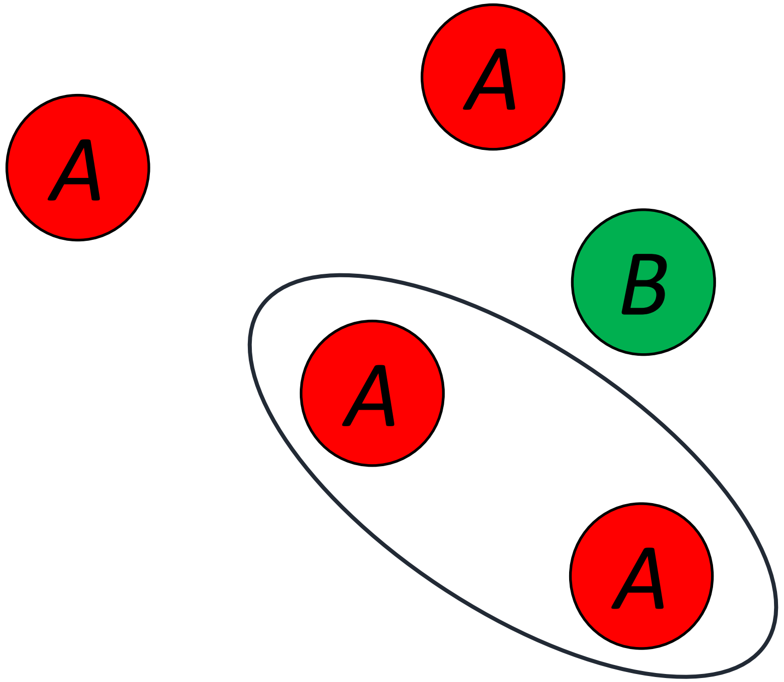
$$
\begin{array}{rccc}
 & A & B & C \\
\mathbf{x} = & (2, & 2, & 0) \\
 & \alpha \Downarrow & & \\
 & (2, & 1, & 1)
\end{array}
$$

What is possible:

Example execution (reaction sequence)

α: $A+B \to A+C$

β: $C \to \boxed{A+A}$



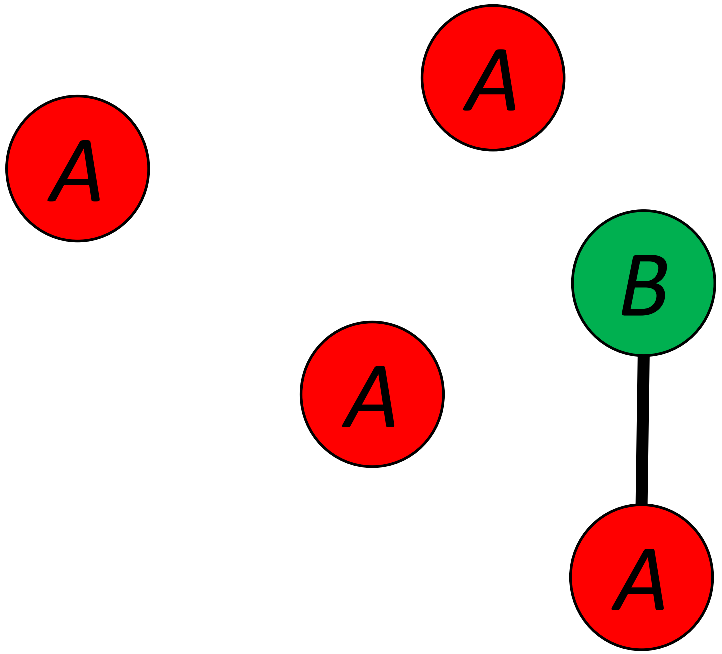
$$
\begin{array}{cccccc}
 & A & B & C & & \\
\mathbf{x} = (& 2, & 2, & 0) & & \\
 & \alpha \Downarrow & & & & \\
(& 2, & 1, & 1) & & \\
 & \beta \Downarrow & & & \searrow \alpha & \\
(& 4, & 1, & 0) & & \cdots
\end{array}
$$

What is possible:

Example execution (reaction sequence)

α: $\boxed{A+B} \to A+C$

β: $C \to A+A$

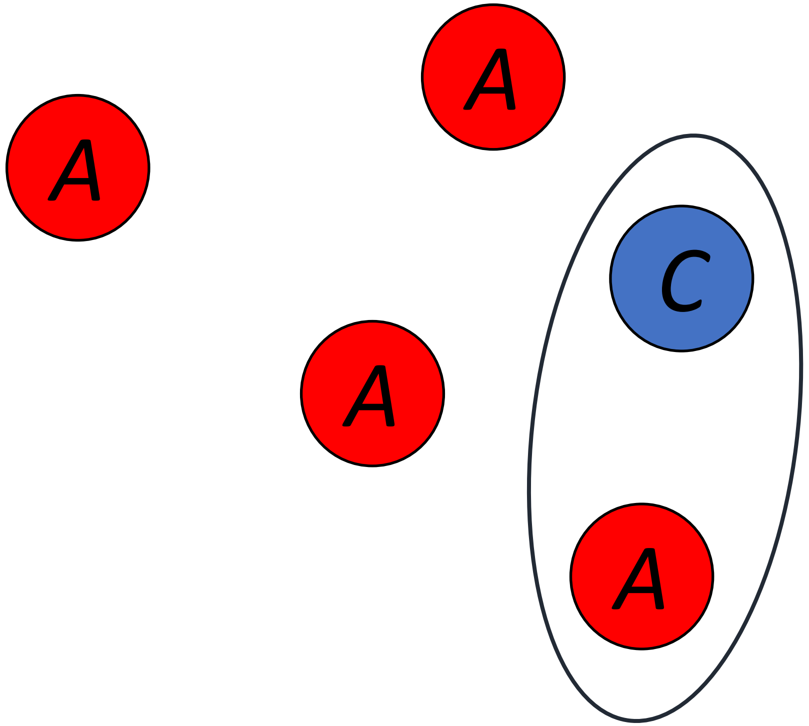


$$
\begin{array}{cccc}
 & A & B & C \\
\mathbf{x} = & (2, & 2, & 0) \\
 & & \alpha \Downarrow & \\
 & (2, & 1, & 1) \\
 & & \beta \Downarrow & \searrow \alpha \\
 & (4, & 1, & 0) \quad \cdots
\end{array}
$$

What is possible:
Example execution (reaction sequence)

α: $A+B \to \boxed{A+C}$

β: $C \to A+A$



$$
\begin{array}{rccc}
 & A & B & C \\
\mathbf{x} = & (2, & 2, & 0) \\
 & & \alpha \Downarrow & \\
 & (2, & 1, & 1) \\
 & & \beta \Downarrow & \quad \text{⇘} \alpha \\
 & (4, & 1, & 0) \quad \cdots \\
 & & \alpha \Downarrow & \\
 & (4, & 0, & 1) \\
 & & \cdots &
\end{array}
$$

Key property of reachability: additivity

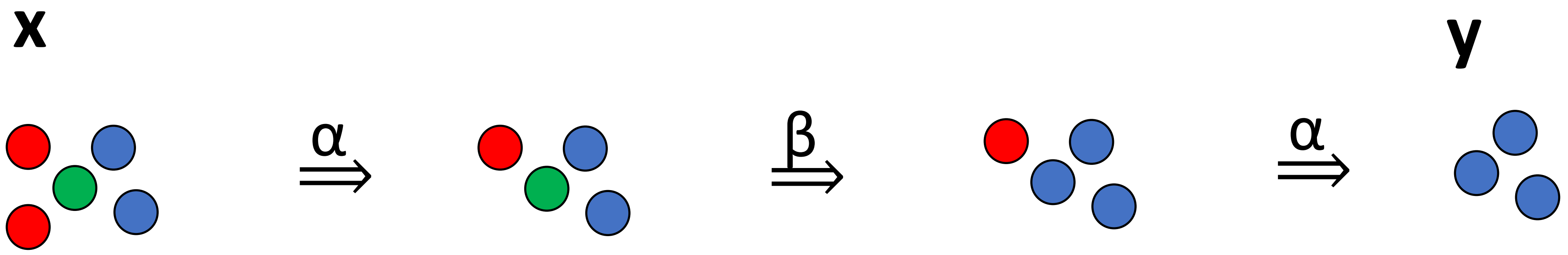
If we can reach from state **x** to **y**, written **x** $\Rightarrow$ **y**, then for all **c** $\in \mathbb{N}^d$,
x+c $\Rightarrow$ **y+c**

The presence of extra molecules (represented by **c**) cannot *prevent* reactions from occurring.

Key property of reachability: <u>additivity</u>

If we can reach from state **x** to **y**, written **x** ⇒ **y**, then for all **c** ∈ $\mathbb{N}^d$, **x+c** ⇒ **y+c**

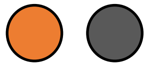
The presence of extra molecules (represented by **c**) cannot *prevent* reactions from occurring.

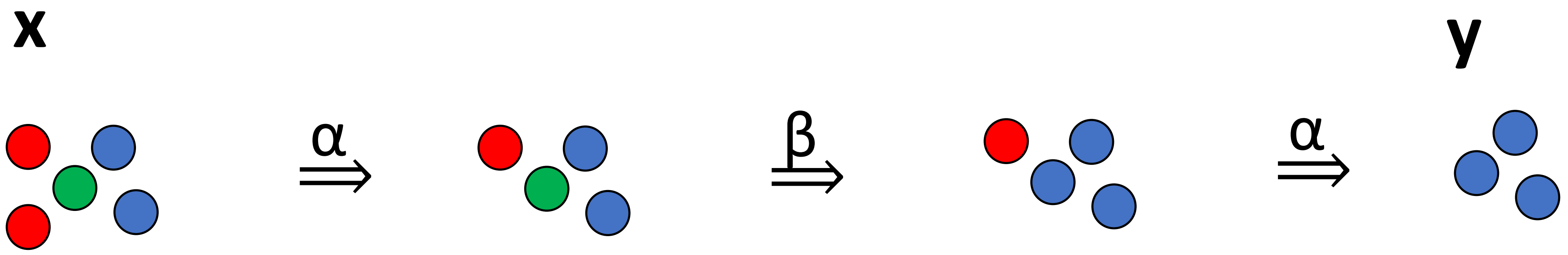


Key property of reachability: <u>additivity</u>

If we can reach from state **x** to **y**, written **x** ⇒ **y**, then for all **c** ∈ $\mathbb{N}^d$, **x**+**c** ⇒ **y**+**c**

The presence of extra molecules (represented by **c**) cannot *prevent* reactions from occurring.

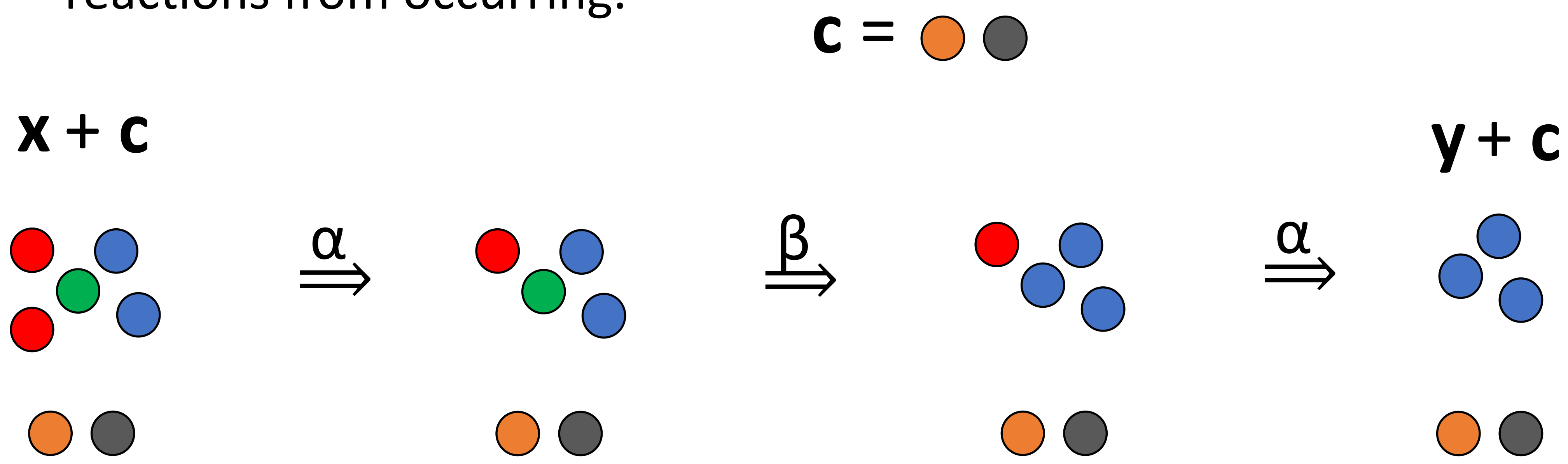
c =




Key property of reachability: <u>additivity</u>

If we can reach from state **x** to **y**, written **x** ⇒ **y**, then for all **c** $\in \mathbb{N}^d$, **x+c** ⇒ **y+c**

The presence of extra molecules (represented by **c**) cannot *prevent* reactions from occurring.



Notation

- For vectors $\mathbf{x},\mathbf{y} \in \mathbb{N}^d$
  - $\mathbf{x} \leqq \mathbf{y}$: $\mathbf{x}(i) \le \mathbf{y}(i)$ for $1 \le i \le d$ $\quad (1,2) \leqq (1,2)$
  - $\mathbf{x} \le \mathbf{y}$: $\mathbf{x} \leqq \mathbf{y}$ and $\mathbf{x} \ne \mathbf{y}$ $\quad (1,2) \le (1,4)$
  - $\mathbf{x} < \mathbf{y}$: $\mathbf{x}(i) < \mathbf{y}(i)$ for $1 \le i \le d$ $\quad (1,2) < (3,4)$

Notation

- For vectors $\mathbf{x},\mathbf{y} \in \mathbb{N}^d$
  - $\mathbf{x} \leqq \mathbf{y}$: $\mathbf{x}(i) \leq \mathbf{y}(i)$ for $1 \leq i \leq d$ — $(1,2) \leqq (1,2)$
  - $\mathbf{x} \leq \mathbf{y}$: $\mathbf{x} \leqq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$ — $(1,2) \leq (1,4)$
  - $\mathbf{x} < \mathbf{y}$: $\mathbf{x}(i) < \mathbf{y}(i)$ for $1 \leq i \leq d$ — $(1,2) < (3,4)$
  - If $\mathbf{x} \geqq \mathbf{0}$, $\mathbf{x}$ is **nonnegative**.
  - If $\mathbf{x} \geq \mathbf{0}$, $\mathbf{x}$ is **semipositive**.
  - If $\mathbf{x} > \mathbf{0}$, $\mathbf{x}$ is **positive**.

Outline

- Formal definition of chemical reaction networks
- **<u>Execution bounded chemical reaction networks and linear potential functions</u>**
- What is “computation” with chemical reactions?
- Limitations of computation with execution bounded chemical reaction networks

Execution bounded CRNs

- Definition: A CRN C is **execution bounded** from state $\mathbf{x}$ if all executions starting at $\mathbf{x}$ are finite.

Execution bounded CRNs

- Definition: A CRN C is **execution bounded** from state **x** if all executions starting at **x** are finite.
- Why prefer execution bounded CRNs?
  - Wet lab implementations of CRNs use up "fuel" to execute reactions; execution bounded CRNs limit the amount of fuel needed
  - Easier to reason about: as long as reactions keep happening, they make "progress" towards reaching a final state.

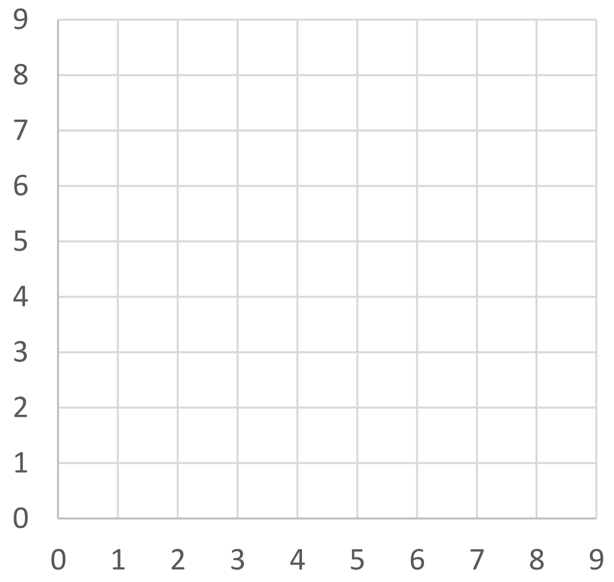
Self-covering executions

Easy Lemma: CRN *C* is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

Self-covering executions

<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

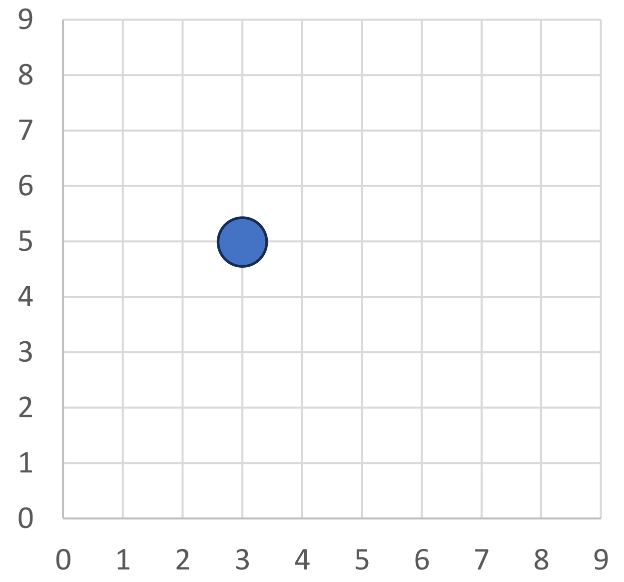
$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

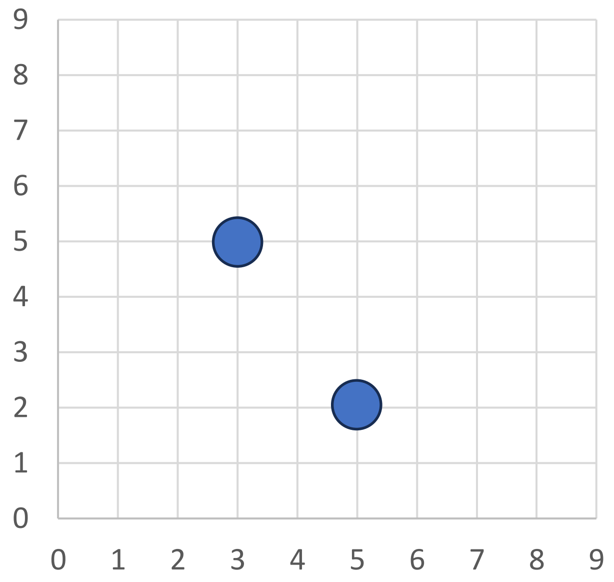
⇒: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

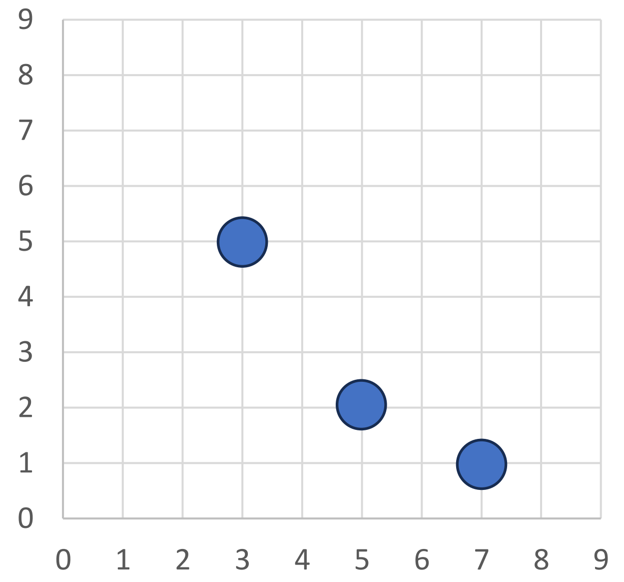
⇒: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

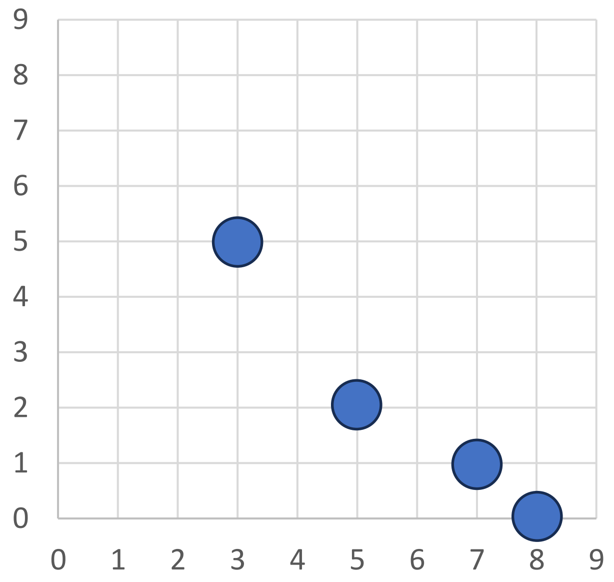
$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

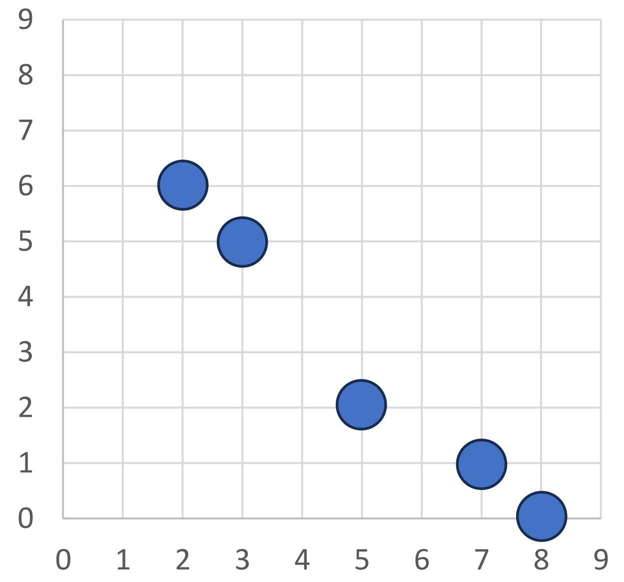
⇒: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

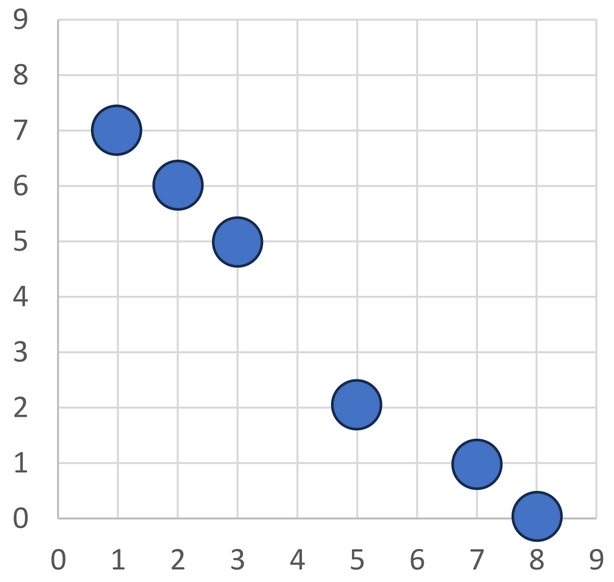
$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

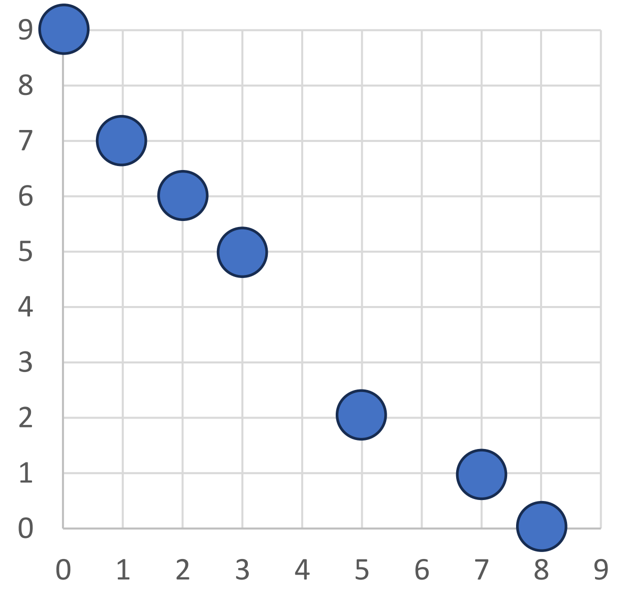
⇒: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

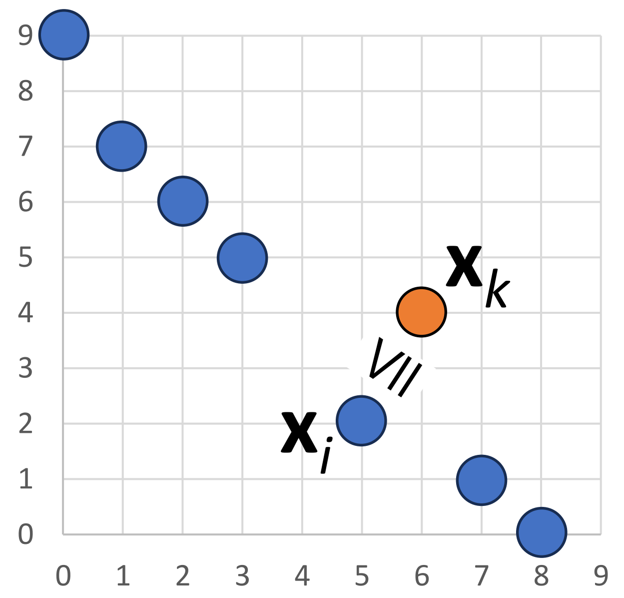
⇒: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



Self-covering executions

<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.

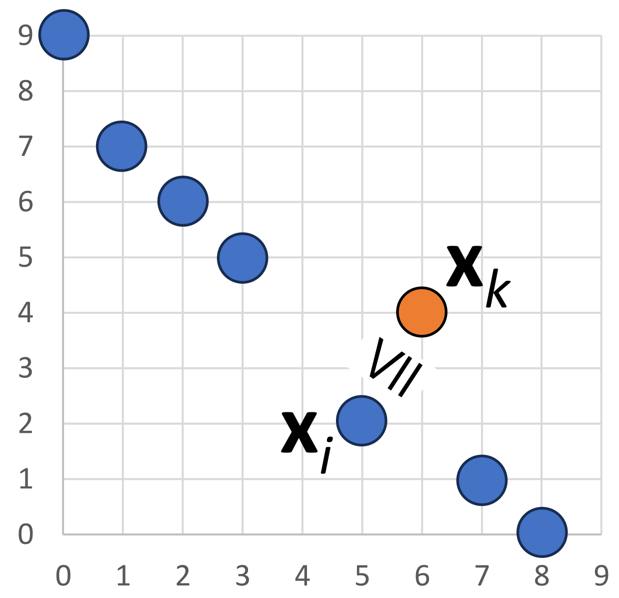


Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.

$\Leftarrow$: If an execution is self-covering, by additivity we can repeat indefinitely the reactions leading from $\mathbf{x}_i$ to $\mathbf{x}_k$, so C is not execution bounded from $\mathbf{x}_0$.

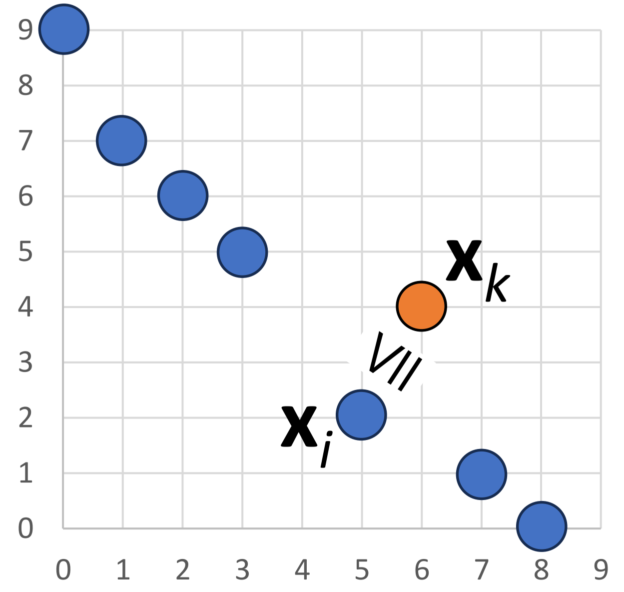


Self-covering executions

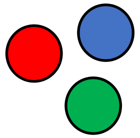
<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.

$\Leftarrow$: If an execution is self-covering, by additivity we can repeat indefinitely the reactions leading from $\mathbf{x}_i$ to $\mathbf{x}_k$, so C is not execution bounded from $\mathbf{x}_0$.



$\mathbf{x}_0$

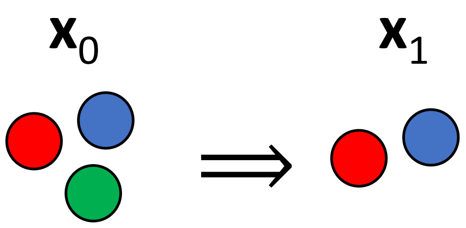


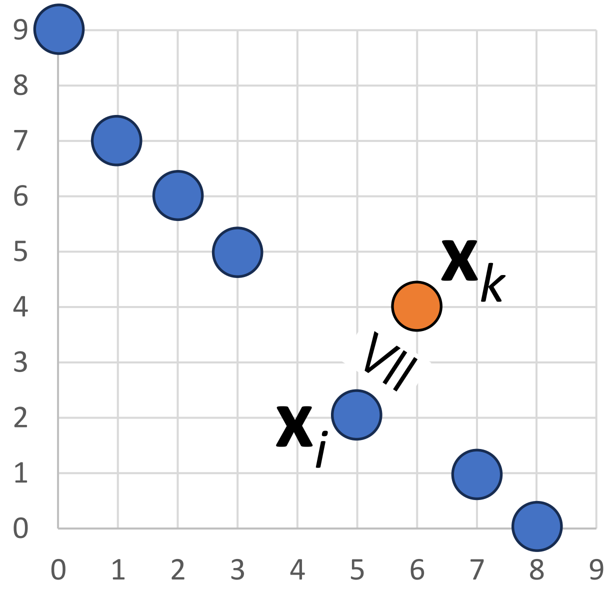
Self-covering executions

<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.

$\Leftarrow$: If an execution is self-covering, by additivity we can repeat indefinitely the reactions leading from $\mathbf{x}_i$ to $\mathbf{x}_k$, so C is not execution bounded from $\mathbf{x}_0$.



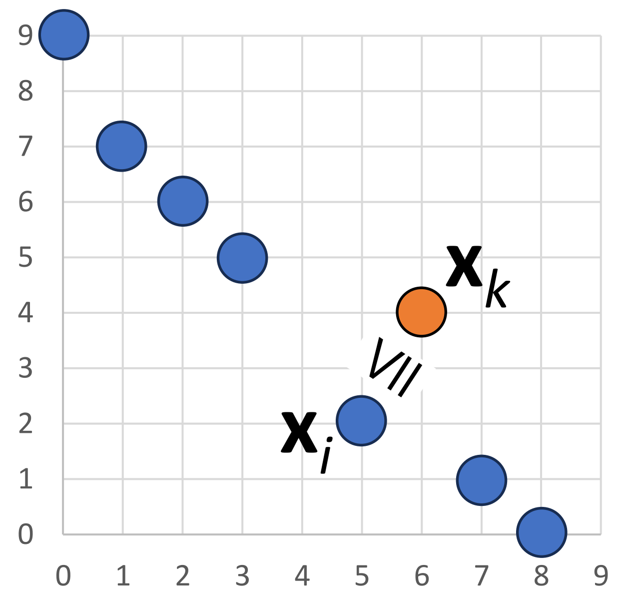


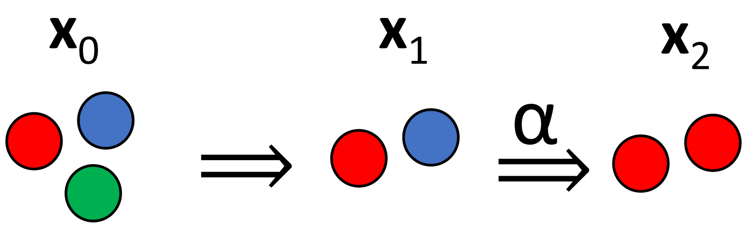
Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

⇒: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.

⇐: If an execution is self-covering, by additivity we can repeat indefinitely the reactions leading from $\mathbf{x}_i$ to $\mathbf{x}_k$, so C is not execution bounded from $\mathbf{x}_0$.

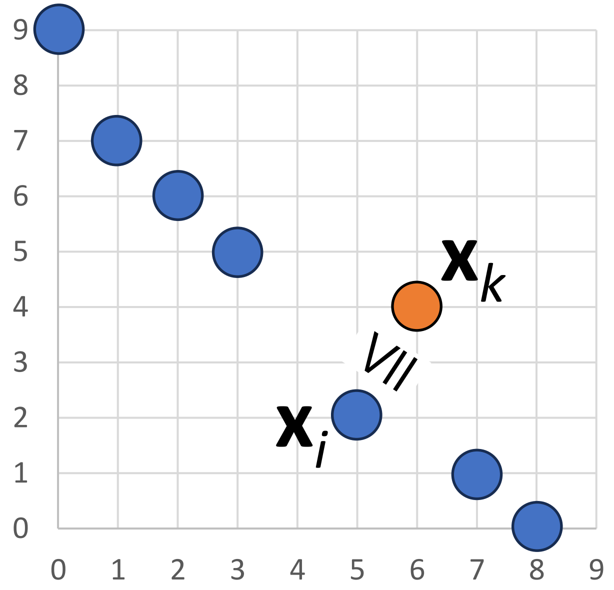




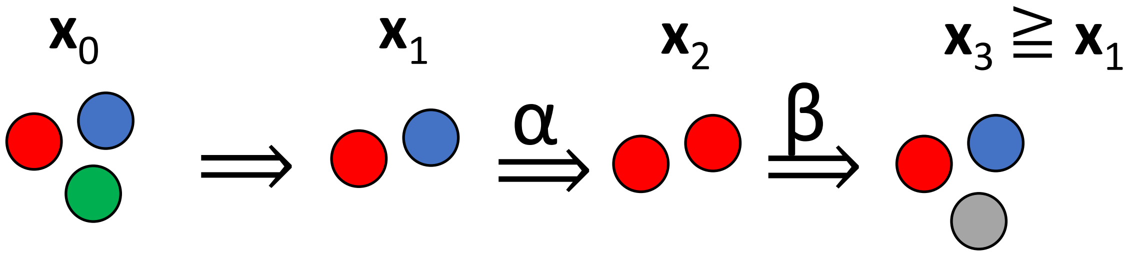
Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.



$\Leftarrow$: If an execution is self-covering, by additivity we can repeat indefinitely the reactions leading from $\mathbf{x}_i$ to $\mathbf{x}_k$, so C is not execution bounded from $\mathbf{x}_0$.

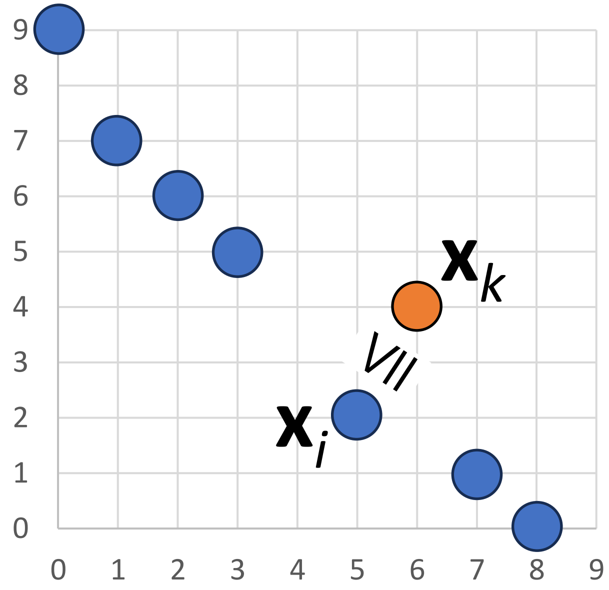


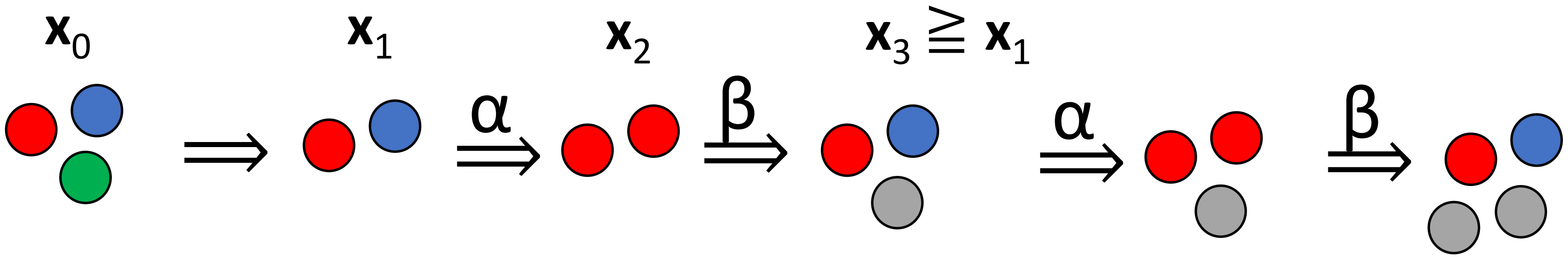
Self-covering executions

Easy Lemma: CRN C is not execution bounded from $\mathbf{x}_0$ if and only if there is an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.

$\Leftarrow$: If an execution is self-covering, by additivity we can repeat indefinitely the reactions leading from $\mathbf{x}_i$ to $\mathbf{x}_k$, so C is not execution bounded from $\mathbf{x}_0$.



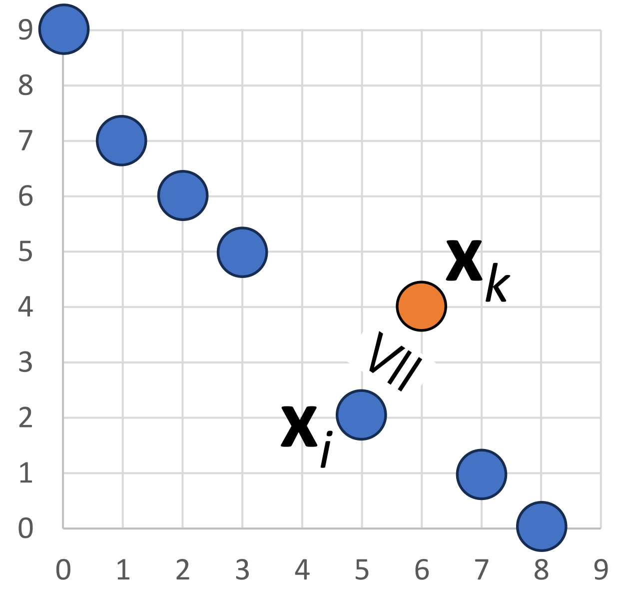


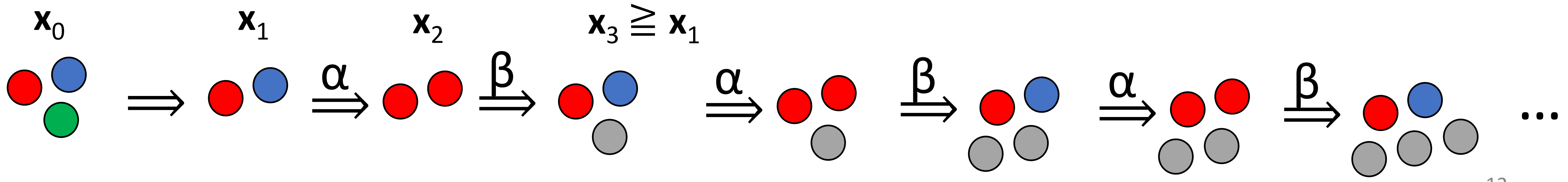
Self-covering executions

<u>Easy Lemma</u>: CRN C is <u>not</u> execution bounded from $\mathbf{x}_0$ if and only if there <u>is</u> an execution $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ that is **self-covering**: $\mathbf{x}_i \leqq \mathbf{x}_k$ for some $i < k$.

$\Rightarrow$: *Dickson's Lemma*: If $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots)$ is any infinite sequence of vectors from $\mathbb{N}^d$, then for some $i < k$, $\mathbf{x}_i \leqq \mathbf{x}_k$. (easy to show by induction on dimension d) So if C has an infinite execution, it is self-covering.

$\Leftarrow$: If an execution is self-covering, by additivity we can repeat indefinitely the reactions leading from $\mathbf{x}_i$ to $\mathbf{x}_k$, so C is not execution bounded from $\mathbf{x}_0$.





Linear potential function

- Definition: $\Phi \colon \mathbb{N}^d \to \mathbb{R}_{\geq 0}$ is a **linear potential function** for a CRN if it is a nonnegative linear function of states that every reaction strictly decreases.

Linear potential function

- Definition: Φ: $\mathbb{N}^d \to \mathbb{R}_{\geq 0}$ is a **linear potential function** for a CRN if it is a nonnegative linear function of states that every reaction strictly decreases.
- Example:
  - $A+A \to B+C$
  - $B+B \to A$
  - A linear potential function $\Phi(\mathbf{x}) = v_A \cdot \mathbf{x}(A) + v_B \cdot \mathbf{x}(B) + v_C \cdot \mathbf{x}(C)$ must satisfy $2v_A > v_B + v_C$ and $2v_B > v_A$ … $v_A = v_B = 1$ and $v_C = 0$ works.

Linear potential function

- Definition: Φ: $\mathbb{N}^d \to \mathbb{R}_{\geq 0}$ is a **linear potential function** for a CRN if it is a nonnegative linear function of states that every reaction strictly decreases.
- Example:
  - $A+A \to B+C$
  - $B+B \to A$
  - A linear potential function $\Phi(\mathbf{x}) = v_A \cdot \mathbf{x}(A) + v_B \cdot \mathbf{x}(B) + v_C \cdot \mathbf{x}(C)$ must satisfy $2v_A > v_B + v_C$ and $2v_B > v_A$ … $v_A = v_B = 1$ and $v_C = 0$ works.
- A coefficient v_S assigns a nonnegative "mass" to species S, and every reaction removes a positive amount of mass from the system.

Linear potential function

- <u>Definition</u>: Φ: $\mathbb{N}^d \to \mathbb{R}_{\geq 0}$ is a **linear potential function** for a CRN if it is a nonnegative linear function of states that every reaction strictly decreases.
- Example:
  - A+$A \to B$+C
  - B+$B \to A$
  - A linear potential function $\Phi(\mathbf{x}) = \mathrm{v}_A \cdot \mathbf{x}(A) + \mathrm{v}_B \cdot \mathbf{x}(B) + \mathrm{v}_C \cdot \mathbf{x}(C)$ must satisfy $2\mathrm{v}_A > \mathrm{v}_B + \mathrm{v}_C$ and $2\mathrm{v}_B > \mathrm{v}_A$ … $\mathrm{v}_A = \mathrm{v}_B = 1$ and $\mathrm{v}_C = 0$ works.
- A coefficient v_S assigns a nonnegative "mass" to species S, and every reaction removes a positive amount of mass from the system.
- By clearing denominators, we can assume each v_S is an integer, so each reaction decreases Φ by at least 1.

Linear potential functions characterize execution bounded CRNs

Theorem: A CRN has a linear potential function if and only if it is execution bounded from every state.

Linear potential functions characterize execution bounded CRNs

Theorem: A CRN has a linear potential function if and only if it is execution bounded from every state.

Forward direction is easy: Since each reaction reduces Φ by at least 1, at most $\Phi(\mathbf{x})$ reactions are possible from any state $\mathbf{x}$.

Key technical tool for reverse direction

Theorem: (Gale 1960) "*Theorem of the Alternative*" (similar to Farkas' Lemma):
Let $\mathbf{M}$ be a matrix. Then exactly one of the following statements is true:

1. There is a vector $\mathbf{u} \geq \mathbf{0}$ such that $\mathbf{Mu} \geqq \mathbf{0}$.
2. There is a vector $\mathbf{v} \geq \mathbf{0}$ such that $\mathbf{vM} < \mathbf{0}$.

[David Gale. The Theory of Linear Economic Models. University of Chicago press, 1960.]

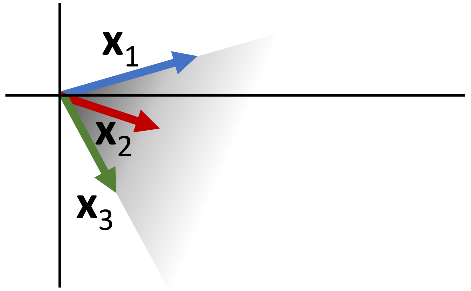
Key technical tool for reverse direction

<u>Theorem:</u> (Gale 1960) "*Theorem of the Alternative*" (similar to Farkas' Lemma):
Let $\mathbf{M}$ be a matrix. Then exactly one of the following statements is true:

1. There is a vector $\mathbf{u} \geq \mathbf{0}$ such that $\mathbf{Mu} \geqq \mathbf{0}$.
2. There is a vector $\mathbf{v} \geq \mathbf{0}$ such that $\mathbf{vM} < \mathbf{0}$.

[David Gale. <u>The Theory of Linear Economic Models</u>. University of Chicago press, 1960.]

1. Either the cone of **M**'s column vectors intersects the nonnegative orthant:



$\mathbf{M} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3]$

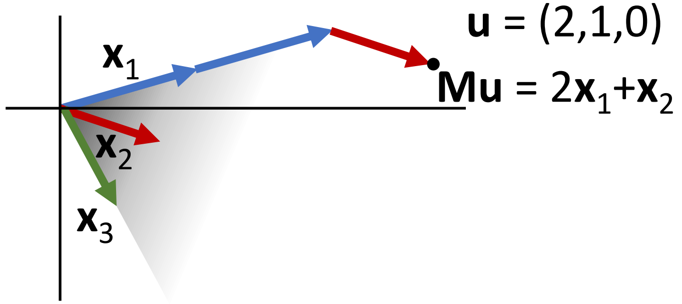
Key technical tool for reverse direction

Theorem: (Gale 1960) "*Theorem of the Alternative*" (similar to Farkas' Lemma):
Let **M** be a matrix. Then exactly one of the following statements is true:

1. There is a vector $\mathbf{u} \geq \mathbf{0}$ such that $\mathbf{Mu} \geqq \mathbf{0}$.
2. There is a vector $\mathbf{v} \geq \mathbf{0}$ such that $\mathbf{vM} < \mathbf{0}$.

[David Gale. The Theory of Linear Economic Models. University of Chicago press, 1960.]

1. Either the cone of **M**'s column vectors intersects the nonnegative orthant:



$\mathbf{M} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3]$

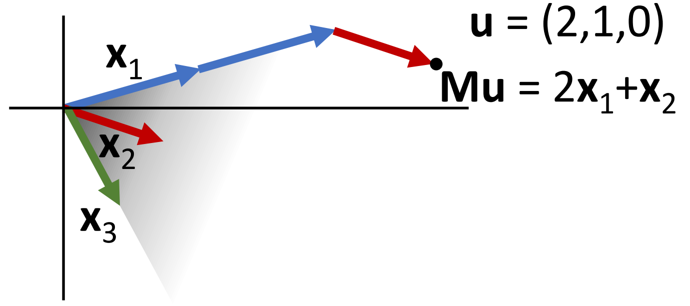
Key technical tool for reverse direction

<u>Theorem:</u> (Gale 1960) "*Theorem of the Alternative*" (similar to Farkas' Lemma):
Let **M** be a matrix. Then exactly one of the following statements is true:

1. There is a vector $\mathbf{u} \geq \mathbf{0}$ such that $\mathbf{Mu} \geqq \mathbf{0}$.
2. There is a vector $\mathbf{v} \geq \mathbf{0}$ such that $\mathbf{vM} < \mathbf{0}$.

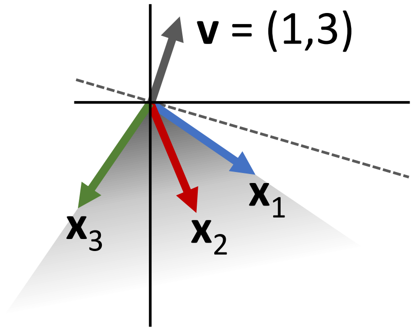
[David Gale. <u>The Theory of Linear Economic Models</u>. University of Chicago press, 1960.]

1. Either the cone of **M**'s column vectors intersects the nonnegative orthant:



$\mathbf{M} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3]$

2. Or it doesn't, and then some hyperplane (dashed line) separates that cone from the nonnegative orthant:



$\mathbf{vM} < \mathbf{0} \Rightarrow$
$(\forall i)\ \mathbf{v} \cdot \mathbf{x}_i < 0$

CRN is execution bounded from every state ⇒ it has a linear potential function

Let **M** be the stoichiometric matrix, e.g.

α: $A \to B + 2C$

β: $3B + C \to A + B + C$

$$\mathbf{M} = \begin{array}{cc} \alpha & \beta \\ \left(\begin{array}{cc} -1 & 1 \\ 1 & -2 \\ 2 & 0 \end{array}\right) \end{array}\begin{array}{c} A \\ B \\ C \end{array}$$

CRN is execution bounded from every state ⇒ it has a linear potential function

Let **M** be the stoichiometric matrix, e.g.

α: $A \rightarrow B + 2C$

β: $3B + C \rightarrow A + B + C$

$$\mathbf{M} = \begin{array}{c} \begin{array}{cc} \alpha & \beta \end{array} \\ \begin{pmatrix} -1 & 1 \\ 1 & -2 \\ 2 & 0 \end{pmatrix} \end{array} \begin{array}{c} A \\ B \\ C \end{array}$$

If **u** = (2,1) is a vector indicating "*do reaction* α *twice and reaction* β *once*", then the vector **Mu** = (–1,0,4) indicates how species counts change.

CRN is execution bounded from every state ⇒ it has a linear potential function

Let **M** be the stoichiometric matrix, e.g.

α: $A \rightarrow B + 2C$

β: $3B + C \rightarrow A + B + C$

$$\mathbf{M} = \begin{array}{cc} \alpha & \beta \\ \left(\begin{array}{cc} -1 & 1 \\ 1 & -2 \\ 2 & 0 \end{array}\right) \end{array} \begin{array}{c} \\ A \\ B \\ C \end{array}$$

If **u** = (2,1) is a vector indicating "*do reaction* α *twice and reaction* β *once*", then the vector **Mu** = (–1,0,4) indicates how species counts change.

Claim: There is no **u** ≥ **0** such that **Mu** ≧ **0**; suppose otherwise. Then from any sufficiently large state **x**, we can execute reactions in **u**, reaching from **x** to **y** = **x** + **Mu**, where **y** ≧ **x**, i.e., a self-covering execution, not possible since the CRN is execution bounded from **x**.

CRN is execution bounded from every state ⇒ it has a linear potential function

Let **M** be the stoichiometric matrix, e.g.

α: $A \rightarrow B + 2C$

β: $3B + C \rightarrow A + B + C$

$$\begin{array}{cc} & \begin{array}{cc} \alpha & \beta \end{array} \\ \mathbf{M} = & \begin{pmatrix} -1 & 1 \\ 1 & -2 \\ 2 & 0 \end{pmatrix} \end{array} \begin{array}{c} \\ A \\ B \\ C \end{array}$$

If **u** = (2,1) is a vector indicating "*do reaction* α *twice and reaction* β *once*", then the vector **Mu** = (–1,0,4) indicates how species counts change.

Claim: There is no **u** ≥ **0** such that **Mu** ≧ **0**; suppose otherwise. Then from any sufficiently large state **x**, we can execute reactions in **u**, reaching from **x** to **y** = **x** + **Mu**, where **y** ≧ **x**, i.e., a self-covering execution, not possible since the CRN is execution bounded from **x**.

Then there is a vector **v** ≥ **0** such that **vM** < **0**. Let **v** be the coefficients of a linear function Φ(**x**) = **v**·**x**. Then **vM** < **0** means each reaction decreases Φ: it is a linear potential function. QED

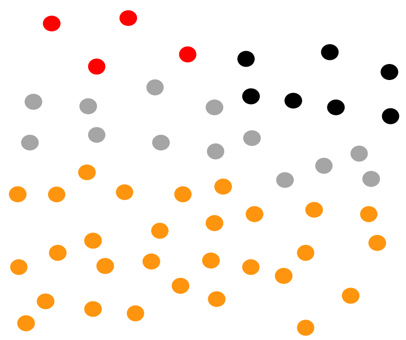
Outline

- Formal definition of chemical reaction networks
- Execution bounded chemical reaction networks and linear potential functions
- <u>**What is “computation” with chemical reactions?**</u>
- Limitations of computation with execution bounded chemical reaction networks

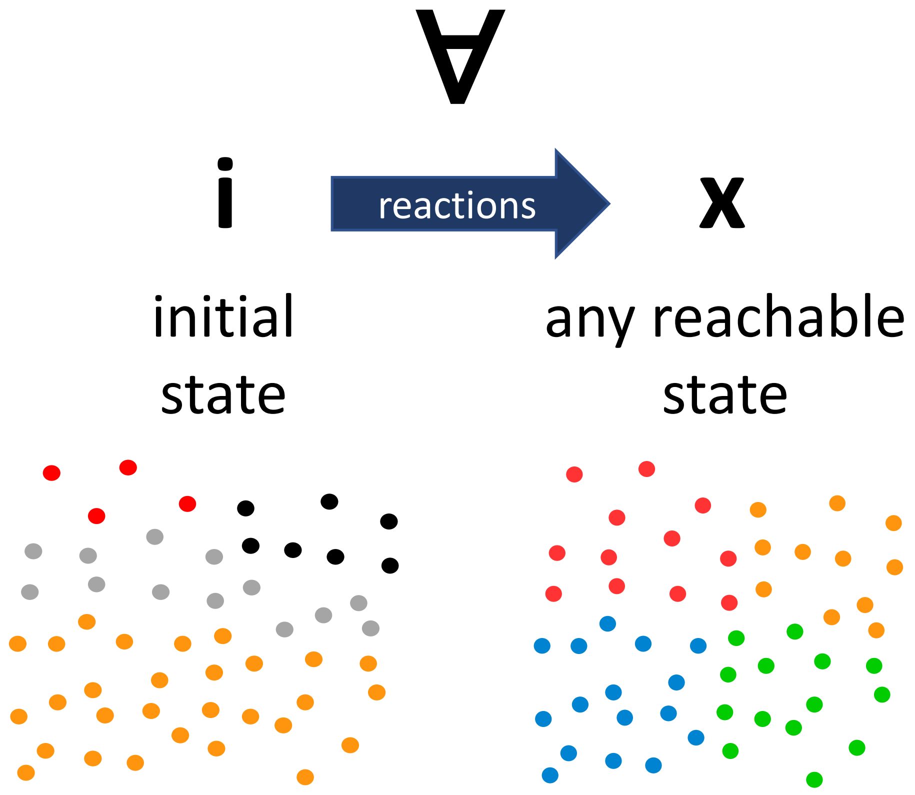
Defining **stable** computation

i

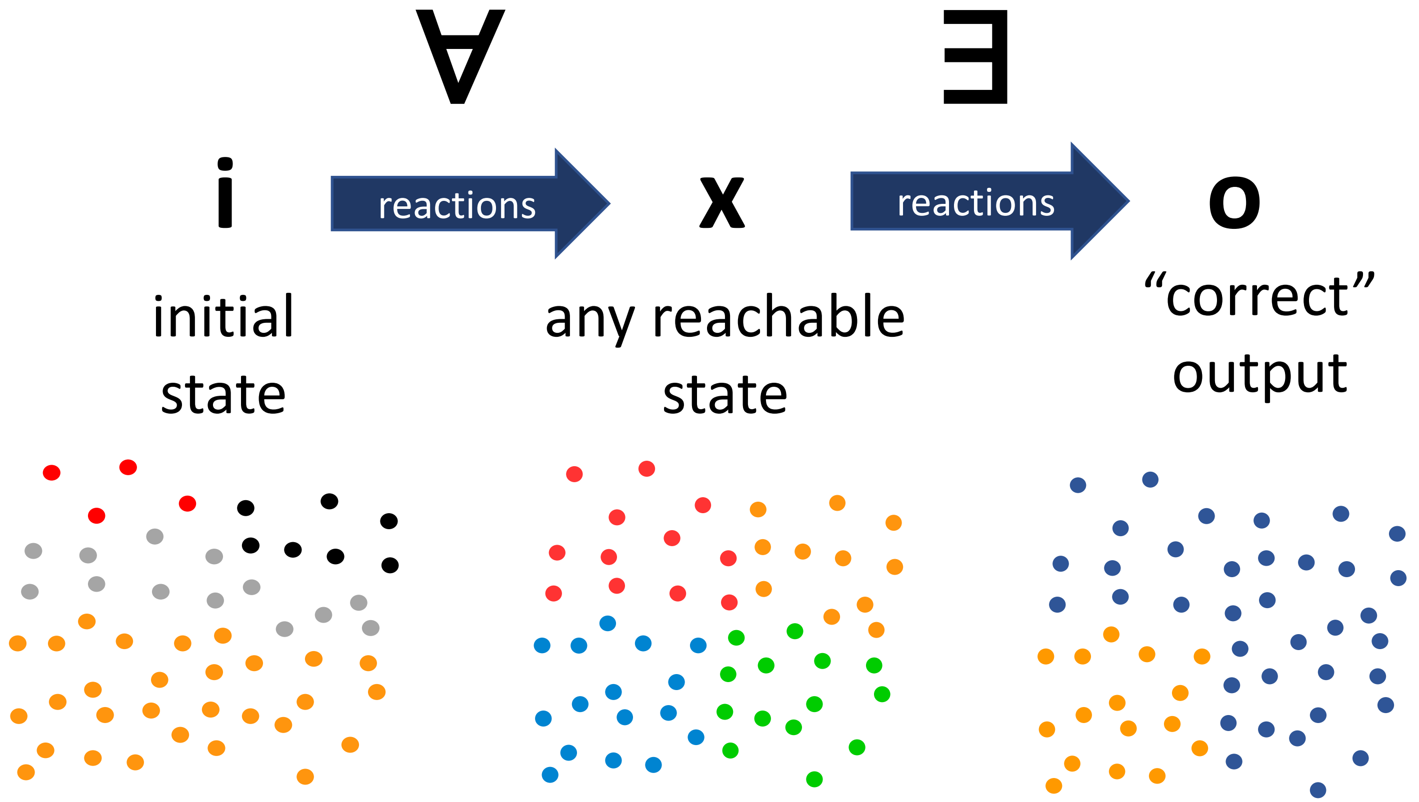
initial
state



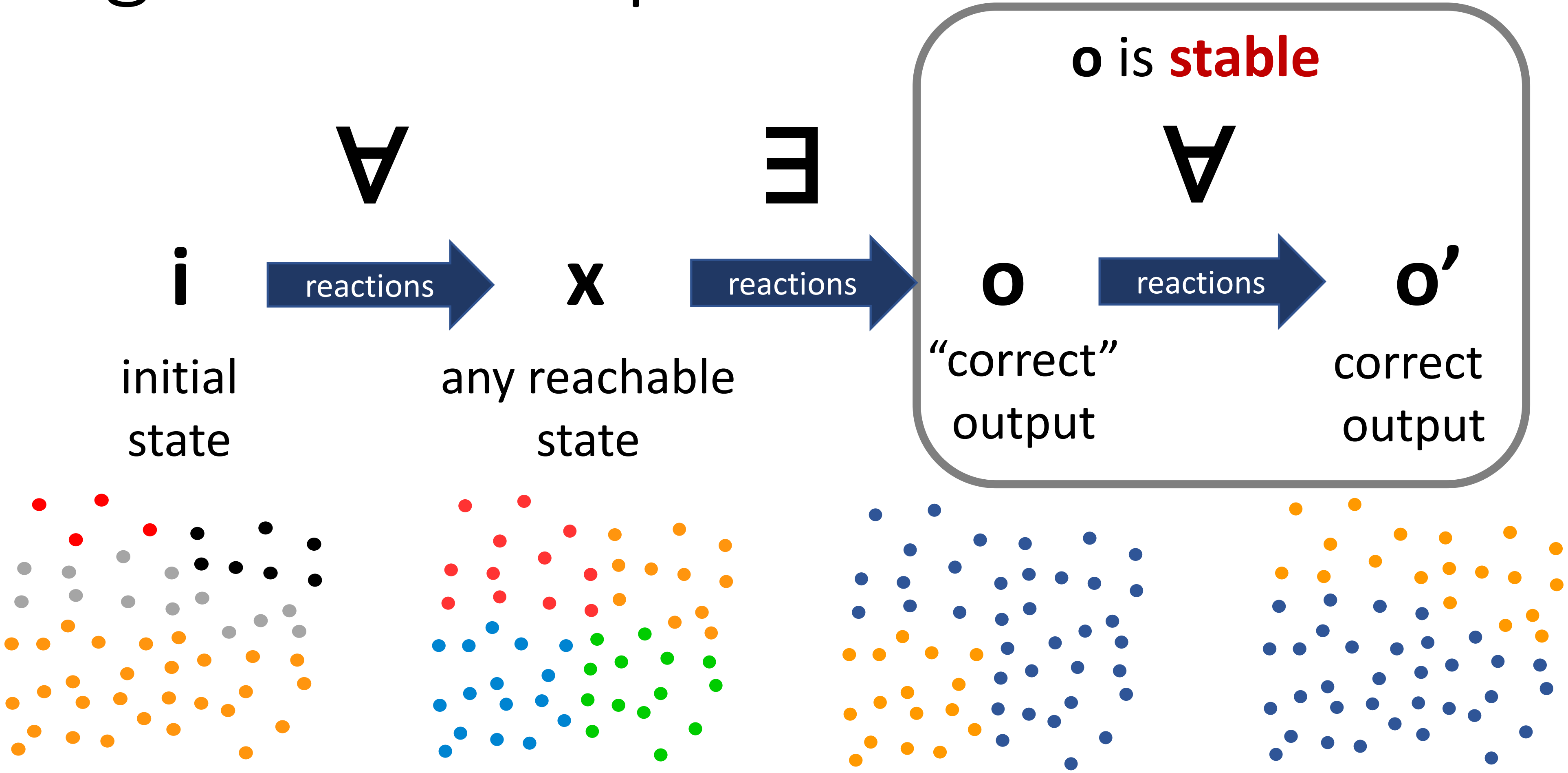
Defining **stable** computation



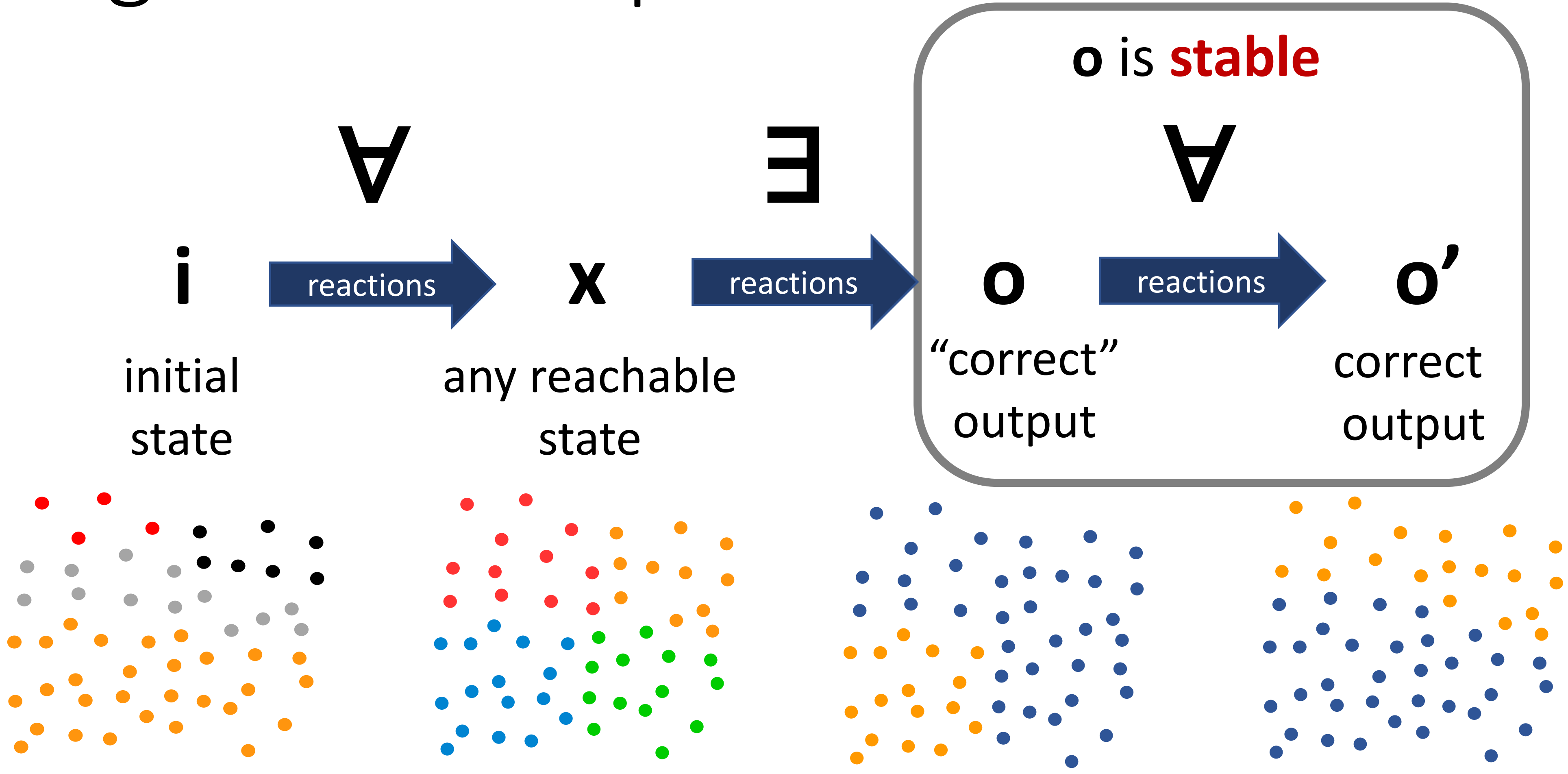
Defining **stable** computation



Defining **stable** computation



Defining **stable** computation



(assuming finite set of reachable states) equivalent to:
The system <u>will</u> reach the correct output with probability 1.

Definition of *predicate* (decision problem) computation

- goal: compute predicate φ: $\mathbb{N}^k \rightarrow \{Y,N\}$, e.g., $\varphi(a,b)=Y \Leftrightarrow a \geq b$

[Angluin, Aspnes, Diamadi, Fischer, Peralta, Computation in networks of passively mobile finite-state sensors, *PODC* 2004]

Definition of *predicate* (decision problem) computation

- goal: compute predicate φ: $\mathbb{N}^k \rightarrow$ {Y,N},   e.g., $\varphi(a,b)$=Y $\Leftrightarrow$ $a \geq b$
- input specification: designate subset $\Sigma \subseteq \Lambda$ as "input" species
  - in valid initial states all molecules are from Σ, e.g., {100 A, 55 B}

[Angluin, Aspnes, Diamadi, Fischer, Peralta, Computation in networks of passively mobile finite-state sensors, *PODC* 2004]

Definition of *predicate* (decision problem) computation

- goal: compute predicate φ: $\mathbb{N}^k \to \{Y,N\}$, e.g., $\varphi(a,b)=Y \Leftrightarrow a \geq b$
- input specification: designate subset $\Sigma \subseteq \Lambda$ as "input" species
  - in valid initial states all molecules are from Σ, e.g., $\{100\ A, 55\ B\}$
- output specification: partition species Λ into "yes" voters Λ_Y and "no" voters Λ_N

[Angluin, Aspnes, Diamadi, Fischer, Peralta, Computation in networks of passively mobile finite-state sensors, *PODC* 2004]

Definition of *predicate* (decision problem) computation

- goal: compute predicate φ: $\mathbb{N}^k \rightarrow \{Y,N\}$,   e.g., $\varphi(a,b)=Y \Leftrightarrow a \geq b$
- input specification: designate subset $\Sigma \subseteq \Lambda$ as "input" species
  - in valid initial states all molecules are from Σ, e.g., $\{100\ A, 55\ B\}$
- output specification: partition species Λ into "yes" voters Λ_Y and "no" voters Λ_N
  - $\psi(\mathbf{o})$ = Y (state **o** outputs "yes") if vote is unanimously yes: $\mathbf{o}(S)>0 \Leftrightarrow S \in \Lambda_Y$
  - $\psi(\mathbf{o})$ = N (state **o** outputs "no") if vote is unanimously no: $\mathbf{o}(S)>0 \Leftrightarrow S \in \Lambda_N$
  - state **o** has undefined output otherwise: $(\exists\ S \in \Lambda_N, S' \in \Lambda_Y)$ $\mathbf{o}(S)>0$ and $\mathbf{o}(S')>0$

[Angluin, Aspnes, Diamadi, Fischer, Peralta, Computation in networks of passively mobile finite-state sensors, *PODC* 2004]

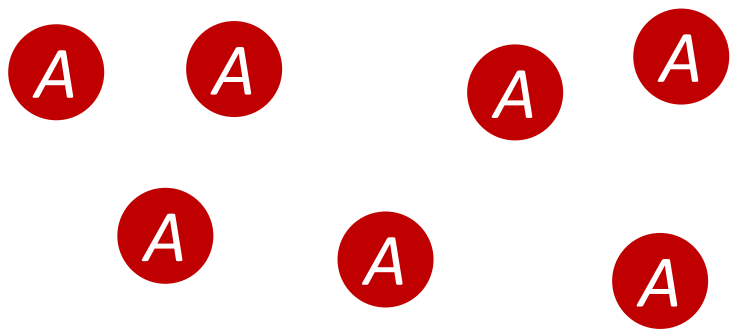
Definition of *predicate* (decision problem) computation

- goal: compute predicate φ: $\mathbb{N}^k \rightarrow \{Y,N\}$,   e.g., $\varphi(a,b)=Y \iff a \geq b$
- input specification: designate subset $\Sigma \subseteq \Lambda$ as "input" species
  - in valid initial states all molecules are from Σ, e.g., {100 *A*, 55 *B*}
- output specification: partition species Λ into "yes" voters Λ_Y and "no" voters Λ_N
  - $\psi(\mathbf{o})$ = Y (state **o** outputs "yes") if vote is unanimously yes: $\mathbf{o}(S)>0 \iff S \in \Lambda_Y$
  - $\psi(\mathbf{o})$ = N (state **o** outputs "no") if vote is unanimously no: $\mathbf{o}(S)>0 \iff S \in \Lambda_N$
  - state **o** has undefined output otherwise: $(\exists\, S \in \Lambda_N, S' \in \Lambda_Y)$ $\mathbf{o}(S)>0$ and $\mathbf{o}(S')>0$
- **o** is stable if $\psi(\mathbf{o}) = \psi(\mathbf{o'})$ for all **o'** reachable from **o**

[Angluin, Aspnes, Diamadi, Fischer, Peralta, Computation in networks of passively mobile finite-state sensors, *PODC* 2004]

Examples of predicate computation

Detection: $\varphi(a,b) = \mathrm{Y} \Leftrightarrow b > 0$

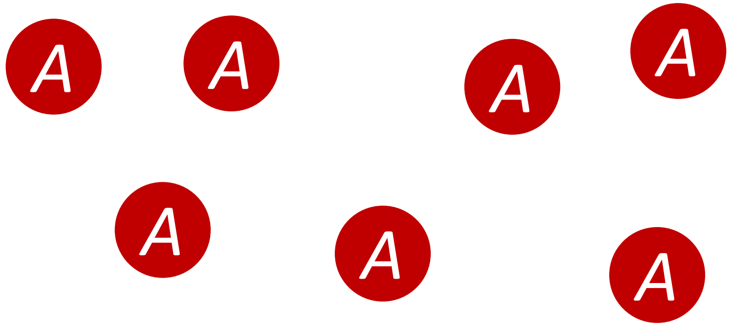


Examples of predicate computation

Detection: $\varphi(a,b) = \mathrm{Y} \Leftrightarrow b > 0$

$$B + A \rightarrow B + B$$

A votes no; *B* votes yes

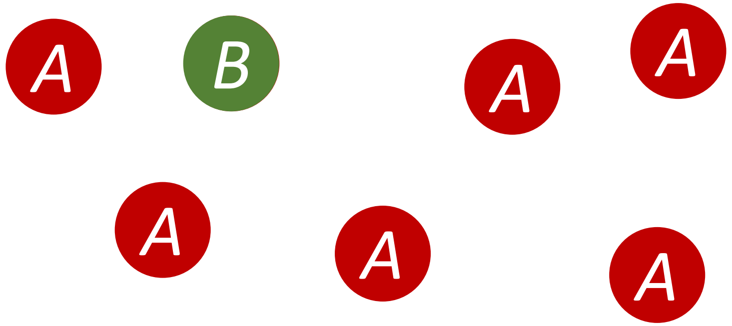


Examples of predicate computation

Detection: $\varphi(a,b) = \mathrm{Y} \Leftrightarrow b > 0$

$$B + A \rightarrow B + B$$

A votes no; *B* votes yes

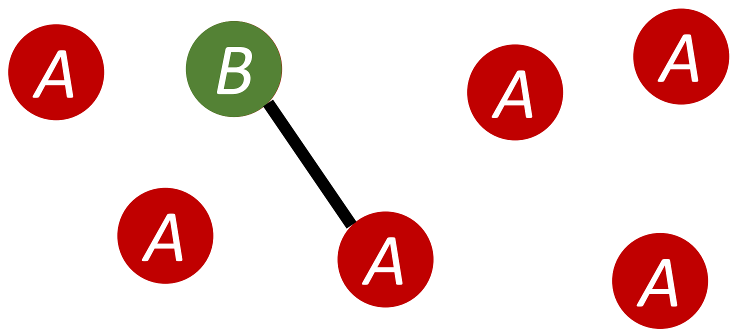


Examples of predicate computation

Detection: $\varphi(a,b) = \mathrm{Y} \Leftrightarrow b > 0$

$$B+A \rightarrow B+B$$

A votes no; B votes yes

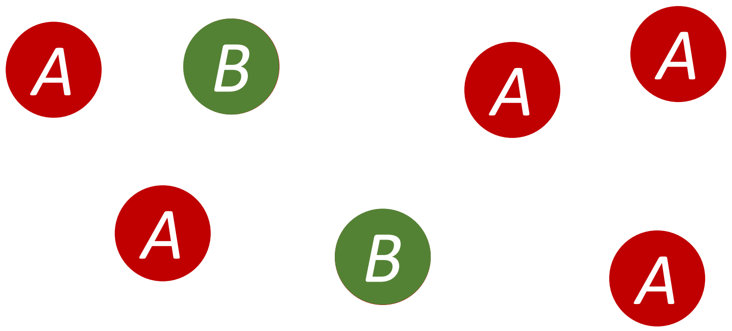


Examples of predicate computation

Detection: $\varphi(a,b) = \mathrm{Y} \Leftrightarrow b > 0$

$$B + A \rightarrow B + B$$

A votes no; *B* votes yes

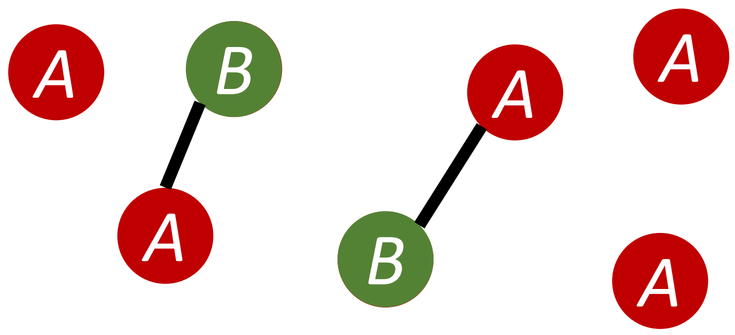


Examples of predicate computation

Detection: $\varphi(a,b) = Y \Leftrightarrow b > 0$

$$B + A \rightarrow B + B$$

A votes no; *B* votes yes

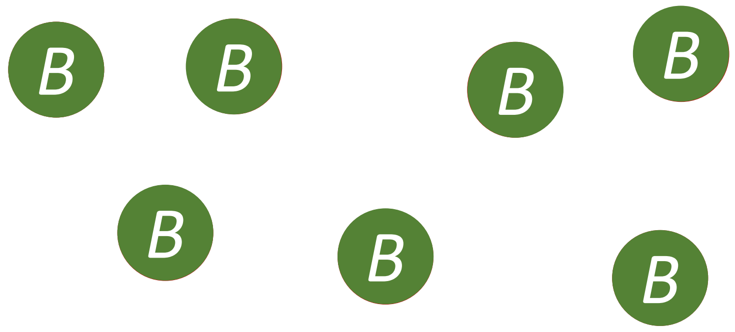


Examples of predicate computation

Detection: $\varphi(a,b) = \text{Y} \Leftrightarrow b > 0$

$$B + A \rightarrow B + B$$

A votes no; *B* votes yes



Examples of predicate computation

Parity: $\varphi(a)$=Y $\Leftrightarrow$ a is odd

Examples of predicate computation

Parity: $\varphi(a)$=Y $\Leftrightarrow$ a is odd

input species A_o (subscript o/e means ODD/EVEN, and capital *A* means it is <u>leader</u>)

Examples of predicate computation

Parity: $\varphi(a)$=Y $\Leftrightarrow$ a is odd

input species A_o (subscript o/e means ODD/EVEN, and capital *A* means it is leader)

$A_o + A_o \rightarrow A_e + a_e$

$A_e + A_e \rightarrow A_e + a_e$

$A_o + A_e \rightarrow A_o + a_o$

two leaders XOR their parity, and one becomes follower

Examples of predicate computation

Parity: $\varphi(a)$=Y ⇔ a is odd

input species A_o (subscript o/e means ODD/EVEN, and capital A means it is <u>leader</u>)

$A_o + A_o \rightarrow A_e + a_e$

$A_e + A_e \rightarrow A_e + a_e$

$A_o + A_e \rightarrow A_o + a_o$

two leaders XOR their parity, and one becomes follower

$A_o + a_e \rightarrow A_o + a_o$

$A_e + a_o \rightarrow A_e + a_e$

leader overwrites bit of follower

Examples of predicate computation

Parity: $\varphi(a)$=Y $\Leftrightarrow$ a is odd

input species A_o (subscript o/e means ODD/EVEN, and capital *A* means it is leader)

$A_o+A_o \rightarrow A_e+a_e$
$A_e+A_e \rightarrow A_e+a_e$
$A_o+A_e \rightarrow A_o+a_o$

two leaders XOR their parity, and one becomes follower

$A_o+a_e \rightarrow A_o+a_o$
$A_e+a_o \rightarrow A_e+a_e$

leader overwrites bit of follower

Not execution bounded!

Limits of stable computation

Theorem: φ: $\mathbb{N}^k \rightarrow$ {Y,N} is stably computable by a CRN if and only if φ is *semilinear*.

semilinear = Boolean combination of threshold and mod predicates:

take weighted sum $s = w_1 \cdot a_1 + \ldots w_k \cdot a_k$ of inputs and ask if

$s >$ constant c?

$s \equiv c \bmod m$ for constants c,m?

$a>b$? $a=b$? a is odd? $a>1$? $a>1$ and b is odd?

NOT $a=b^2$? a is a power of 2? a is prime?

[Angluin, Aspnes, Diamadi, Fischer, Peralta, Computation in networks of passively mobile finite-state sensors, *PODC* 2004]
[Angluin, Aspnes, Eisenstat, Stably computable predicates are semilinear, *PODC* 2006]

Outline

- Formal definition of chemical reaction networks
- Execution bounded chemical reaction networks and linear potential functions
- What is “computation” with chemical reactions?
- <u>**Limitations of computation with execution bounded chemical reaction networks**</u>

Noncollapsing CRNs

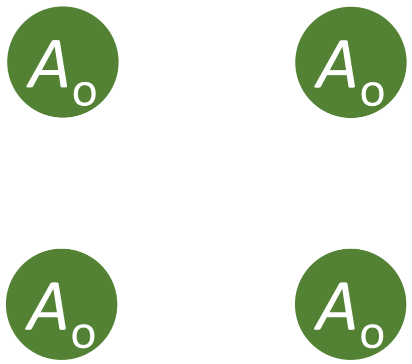
Definition: A CRN is **noncollapsing** if $\lim_{n \to \infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Noncollapsing CRNs

Definition: A CRN is **noncollapsing** if $\lim_{n\to\infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Rules out CRNs such as

$$A_o + A_o \to A_e$$
$$A_e + A_e \to A_e$$
$$A_o + A_e \to A_o$$



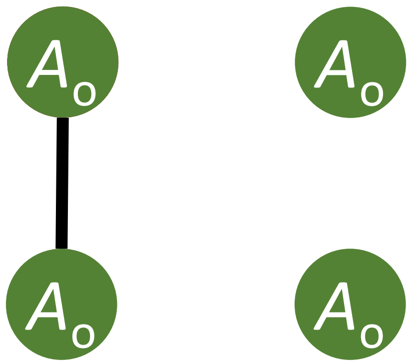
which computes parity but always ends up with a single voter.

Noncollapsing CRNs

<u>Definition</u>: A CRN is **noncollapsing** if $\lim_{n\to\infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Rules out CRNs such as

$$A_o + A_o \to A_e$$
$$A_e + A_e \to A_e$$
$$A_o + A_e \to A_o$$



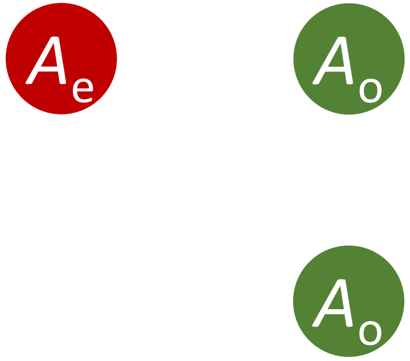
which computes parity but always ends up with a single voter.

Noncollapsing CRNs

Definition: A CRN is **noncollapsing** if $\lim_{n\to\infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Rules out CRNs such as

$$A_o + A_o \to A_e$$
$$A_e + A_e \to A_e$$
$$A_o + A_e \to A_o$$



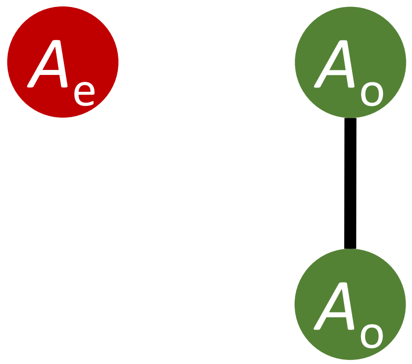
which computes parity but always ends up with a single voter.

Noncollapsing CRNs

Definition: A CRN is **noncollapsing** if $\lim_{n\to\infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Rules out CRNs such as

$A_o + A_o \to A_e$
$A_e + A_e \to A_e$
$A_o + A_e \to A_o$



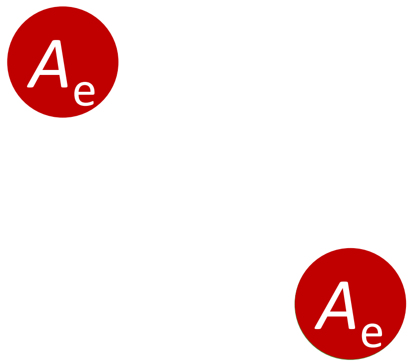
which computes parity but always ends up with a single voter.

Noncollapsing CRNs

Definition: A CRN is **noncollapsing** if $\lim_{n\to\infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Rules out CRNs such as

$$A_o + A_o \rightarrow A_e$$
$$A_e + A_e \rightarrow A_e$$
$$A_o + A_e \rightarrow A_o$$



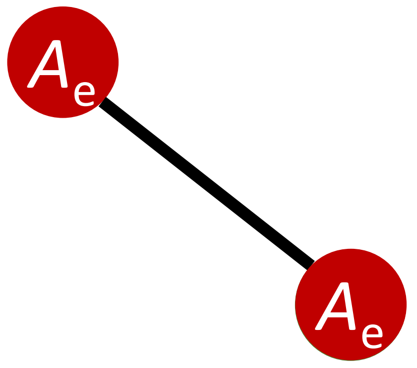
which computes parity but always ends up with a single voter.

Noncollapsing CRNs

Definition: A CRN is **noncollapsing** if $\lim_{n\to\infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Rules out CRNs such as

$A_o + A_o \to A_e$
$A_e + A_e \to A_e$
$A_o + A_e \to A_o$



which computes parity but always ends up with a single voter.

Noncollapsing CRNs

Definition: A CRN is **noncollapsing** if $\lim_{n\to\infty} s(n) = \infty$, where $s(n)$ = size of smallest stable state reachable from any initial state of size n.

Rules out CRNs such as

$$A_o + A_o \to A_e$$
$$A_e + A_e \to A_e$$
$$A_o + A_e \to A_o$$

which computes parity but always ends up with a single voter.

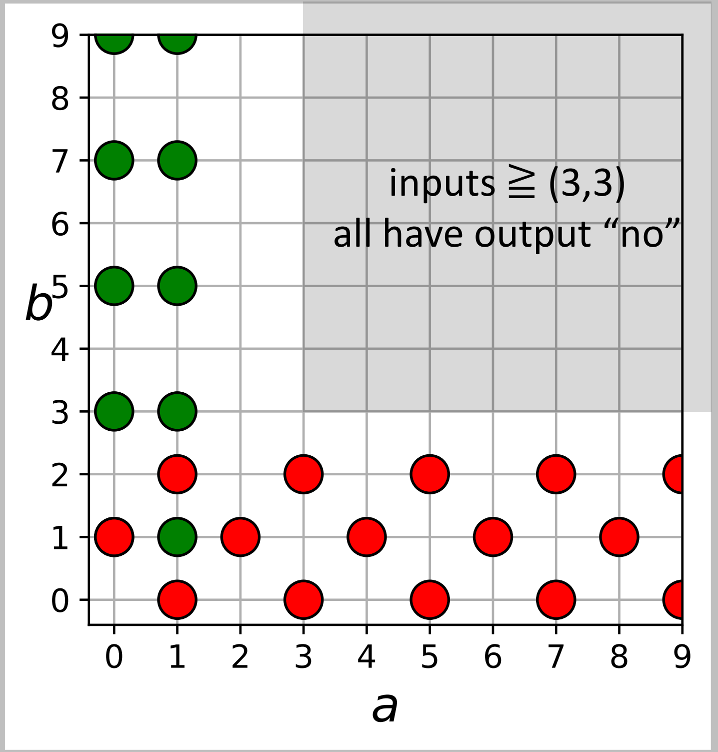
Eventually constant predicates

Definition: A φ: $\mathbb{N}^k \rightarrow$ {Y,N} is **eventually constant** if, for some $c \in \mathbb{N}$, $\varphi(\mathbf{x})$ is constant on all inputs $\mathbf{x} \geqq (c,c,\ldots,c)$.

Eventually constant predicates

<u>Definition</u>: A φ: $\mathbb{N}^k \rightarrow \{Y,N\}$ is **eventually constant** if, for some $c \in \mathbb{N}$, $\varphi(\mathbf{x})$ is constant on all inputs $\mathbf{x} \geqq (c,c,\ldots,c)$.

Example of eventually constant predicate:
$a < 2$ and b is odd, or $b < 3$ and $a+b$ is odd



Non-eventually constant predicates:

majority ($a \geq b$?)

parity (a is odd?)

equality ($a=b$?)

and most anything interesting.

Limitations of execution bounded CRNs

Theorem: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

Limitations of execution bounded CRNs

Theorem: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

Proof: complex.

Proof that such CRNs cannot compute parity (a is odd?):

Limitations of execution bounded CRNs

Theorem: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

Proof: complex.

Proof that such CRNs cannot compute parity (a is odd?):

1. Start with $\{A\}$, CRN can reach to stable "yes" state $\mathbf{s}_1$.

{A}

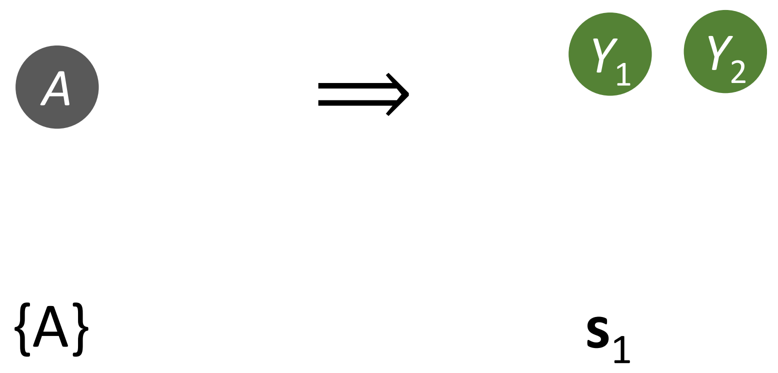
Limitations of execution bounded CRNs

<u>Theorem</u>: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

<u>Proof</u>: complex.

<u>Proof that such CRNs cannot compute parity (a is odd?)</u>:

1. Start with $\{A\}$, CRN can reach to stable "yes" state $\mathbf{s}_1$.



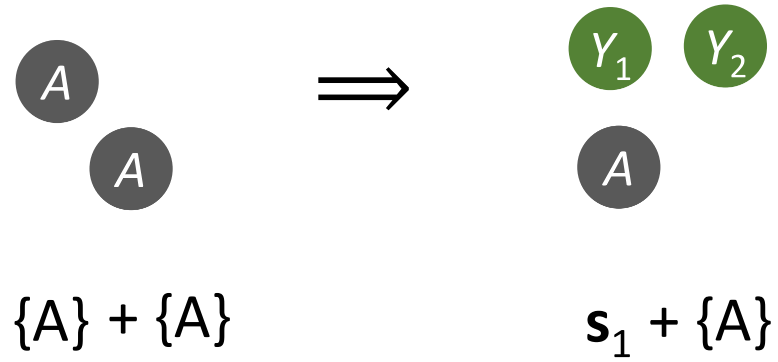
Limitations of execution bounded CRNs

Theorem: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

Proof: complex.

Proof that such CRNs cannot compute parity (a is odd?):

1. Start with $\{A\}$, CRN can reach to stable "yes" state $\mathbf{s}_1$.
2. Add 1 A. The state $\mathbf{s}_1+\{A\}$ is reachable from $\{2A\}$, so the CRN can reach from there to a stable "no" state $\mathbf{s}_2$.



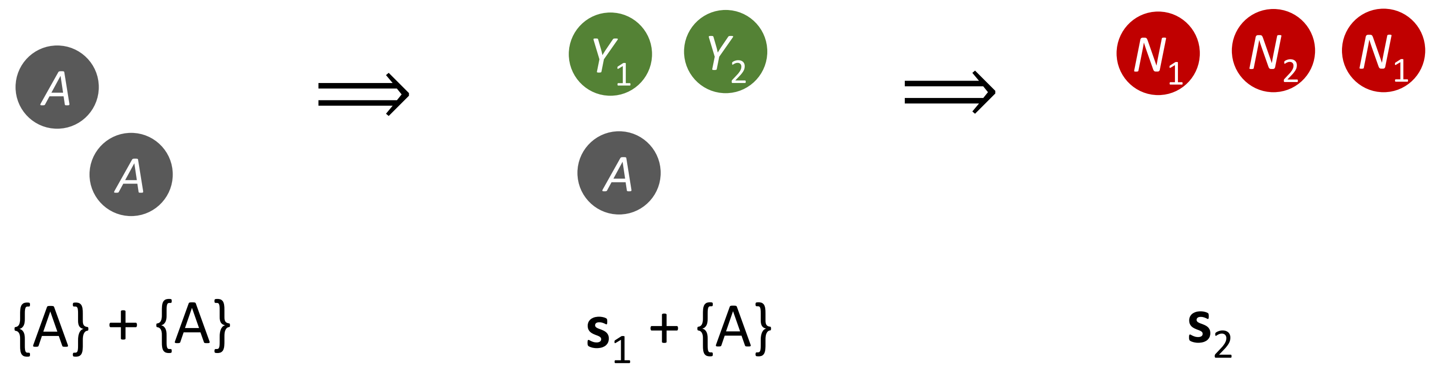
Limitations of execution bounded CRNs

Theorem: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

Proof: complex.

Proof that such CRNs cannot compute parity (a is odd?):

1. Start with $\{A\}$, CRN can reach to stable "yes" state $\mathbf{s}_1$.
2. Add 1 A. The state $\mathbf{s}_1+\{A\}$ is reachable from $\{2A\}$, so the CRN can reach from there to a stable "no" state $\mathbf{s}_2$.



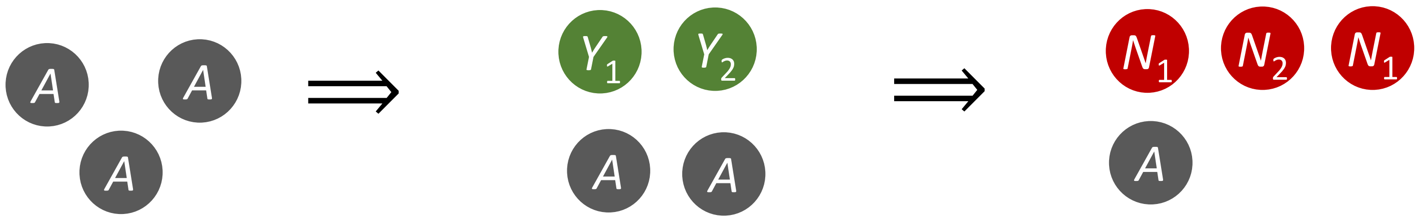
Limitations of execution bounded CRNs

Theorem: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

Proof: complex.

Proof that such CRNs cannot compute parity (a is odd?):

1. Start with $\{A\}$, CRN can reach to stable "yes" state $\mathbf{s}_1$.
2. Add 1 A. The state $\mathbf{s}_1+\{A\}$ is reachable from $\{2A\}$, so the CRN can reach from there to a stable "no" state $\mathbf{s}_2$.
3. Add 1 A. The state $\mathbf{s}_2+\{A\}$ is reachable from $\{3A\}$, so the CRN can reach from there to a stable "yes" state $\mathbf{s}_3$.
4. …



$\{A\} + \{A\} + \{A\}$ $\qquad$ $\mathbf{s}_1 + \{A\} + \{A\}$ $\qquad$ $\mathbf{s}_2 + \{A\}$

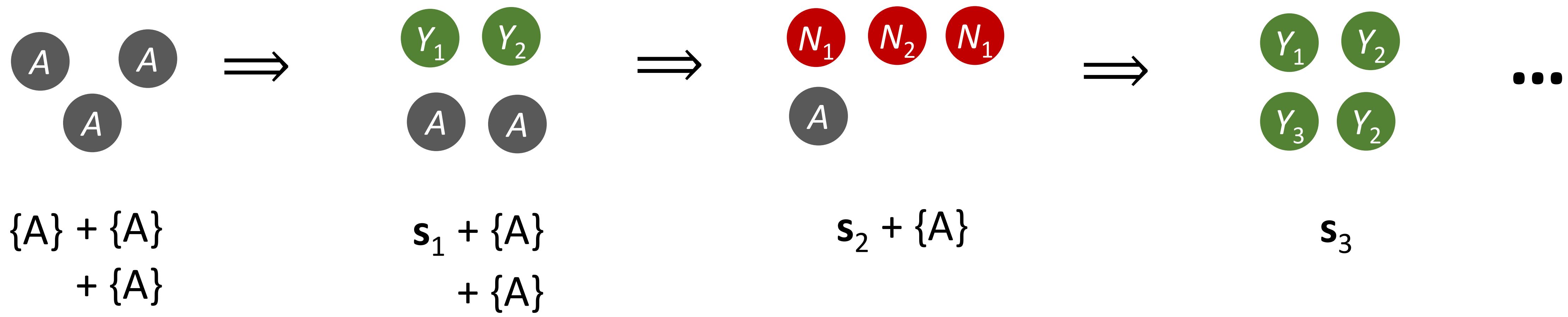
Limitations of execution bounded CRNs

<u>Theorem</u>: If a CRN stably computing φ is noncollapsing and execution bounded from every input state, then φ is eventually constant.

<u>Proof</u>: complex.

<u>Proof that such CRNs cannot compute parity (a is odd?)</u>:

1. Start with $\{A\}$, CRN can reach to stable "yes" state $\mathbf{s}_1$.
2. Add 1 A. The state $\mathbf{s}_1+\{A\}$ is reachable from $\{2A\}$, so the CRN can reach from there to a stable "no" state $\mathbf{s}_2$.
3. Add 1 A. The state $\mathbf{s}_2+\{A\}$ is reachable from $\{3A\}$, so the CRN can reach from there to a stable "yes" state $\mathbf{s}_3$.
4. …



Limitations of execution bounded CRNs

- Since CRN is execution bounded from all states, it has a linear potential function Φ.

Limitations of execution bounded CRNs

- Since CRN is execution bounded from all states, it has a linear potential function Φ.
- Adding $\{A\}$ to $\mathbf{s}_i$ <u>increases</u> Φ by the constant $\Phi(\{A\})$.

Limitations of execution bounded CRNs

- Since CRN is execution bounded from all states, it has a linear potential function Φ.
- Adding $\{A\}$ to $\mathbf{s}_i$ increases Φ by the constant Φ($\{A\}$).
- To get from $\mathbf{s}_i+\{A\}$ to $\mathbf{s}_{i+1}$, since $\lim_{i\to\infty} |\mathbf{s}_i| = \infty$ (noncollapsing), we must execute increasingly more reactions as $i \to \infty$, which all decrease Φ.
  - Key reason: all species vote, so all molecules in $\mathbf{s}_i$ must be removed to switch the output.

Limitations of execution bounded CRNs

- Since CRN is execution bounded from all states, it has a linear potential function Φ.
- Adding $\{A\}$ to $\mathbf{s}_i$ increases Φ by the constant Φ($\{A\}$).
- To get from $\mathbf{s}_i+\{A\}$ to $\mathbf{s}_{i+1}$, since $\lim_{i\to\infty} |\mathbf{s}_i| = \infty$ (noncollapsing), we must execute increasingly more reactions as $i \to \infty$, which all decrease Φ.
  - Key reason: all species vote, so all molecules in $\mathbf{s}_i$ must be removed to switch the output.
- After some i, the net change in Φ, in going from $\mathbf{s}_i$ to $\mathbf{s}_i+\{A\}$ to $\mathbf{s}_{i+1}$, is negative.

Limitations of execution bounded CRNs

- Since CRN is execution bounded from all states, it has a linear potential function Φ.
- Adding $\{A\}$ to $\mathbf{s}_i$ increases Φ by the constant $\Phi(\{A\})$.
- To get from $\mathbf{s}_i+\{A\}$ to $\mathbf{s}_{i+1}$, since $\lim_{i \to \infty} |\mathbf{s}_i| = \infty$ (noncollapsing), we must execute increasingly more reactions as $i \to \infty$, which all decrease Φ.
  - Key reason: all species vote, so all molecules in $\mathbf{s}_i$ must be removed to switch the output.
- After some i, the net change in Φ, in going from $\mathbf{s}_i$ to $\mathbf{s}_i+\{A\}$ to $\mathbf{s}_{i+1}$, is negative.
- Since Φ is nonnegative, at some point we cannot continue. QED

Are execution bounded CRNs good for any computation?

- Yes! Execution bounded CRNs *can* stably compute all semilinear predicates if:

Are execution bounded CRNs good for any computation?

- Yes! Execution bounded CRNs *can* stably compute all semilinear predicates if:
  - Not all species are required to vote, and

Are execution bounded CRNs good for any computation?

- Yes! Execution bounded CRNs *can* stably compute all semilinear predicates if:
  - Not all species are required to vote, and
  - We can start with an "initial leader", e.g., to compute majority ($a \geq b$?), start in initial state {1 L, a A, b B}… these are execution bounded from such states, but not from states with multiple leaders.

Are execution bounded CRNs good for any computation?

- Yes! Execution bounded CRNs *can* stably compute all semilinear predicates if:
  - Not all species are required to vote, and
  - We can start with an “initial leader”, e.g., to compute majority ($a \geq b$?), start in initial state {1 L, a A, b B}… these are execution bounded from such states, but not from states with multiple leaders.
  - Or if all species are required to vote, but the CRN can be collapsing.

Are execution bounded CRNs good for any computation?

- Yes! Execution bounded CRNs *can* stably compute all semilinear predicates if:
  - Not all species are required to vote, and
  - We can start with an "initial leader", e.g., to compute majority ($a \geq b$?), start in initial state {1 L, a A, b B}… these are execution bounded from such states, but not from states with multiple leaders.
  - Or if all species are required to vote, but the CRN can be collapsing.
- Those CRNs take expected time $O(n \log n)$ to converge, whereas non-execution bounded (and leader-driven) CRNs can stably compute all semilinear predicates in expected time $O(\text{polylog}(n))$.

Are execution bounded CRNs good for any computation?

- Yes! Execution bounded CRNs *can* stably compute all semilinear predicates if:
  - Not all species are required to vote, and
  - We can start with an "initial leader", e.g., to compute majority ($a \geq b$?), start in initial state {1 L, a A, b B}… these are execution bounded from such states, but not from states with multiple leaders.
  - Or if all species are required to vote, but the CRN can be collapsing.
- Those CRNs take expected time $O(n \log n)$ to converge, whereas non-execution bounded (and leader-driven) CRNs can stably compute all semilinear predicates in expected time $O(\text{polylog}(n))$.
- <u>Conjecture</u>: Any execution bounded CRN takes at least $\Omega(n)$ expected time to stably compute any non-eventually-constant predicate.

Thank you!

Questions?