Natural computing models of unusual complexity

Antonio E. Porreca
Aix-Marseille Université, Lab. Informatique et Systèmes, équipe CANA
https://aeporreca.org

Outline

- The first and second machine classes
- Membrane computing
- Complexity theory of membrane systems
- Communication topologies and their role

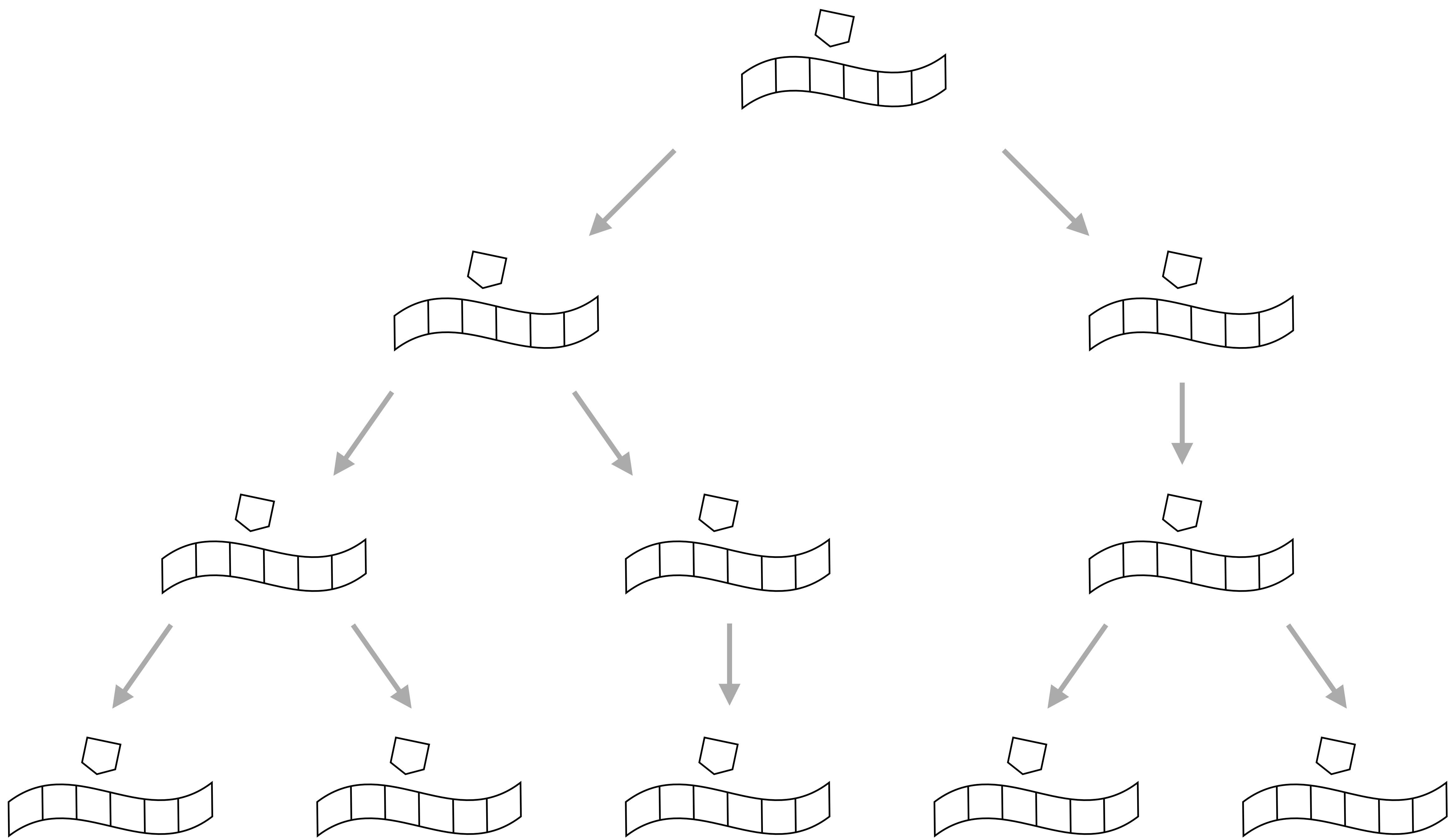
The first machine class and **P**

- The deterministic Turing machine and all models that simulate and are simulated by it efficiently
  - Random access machines with arithmetic operations + and –
  - Cellular automata with finite initial configuration

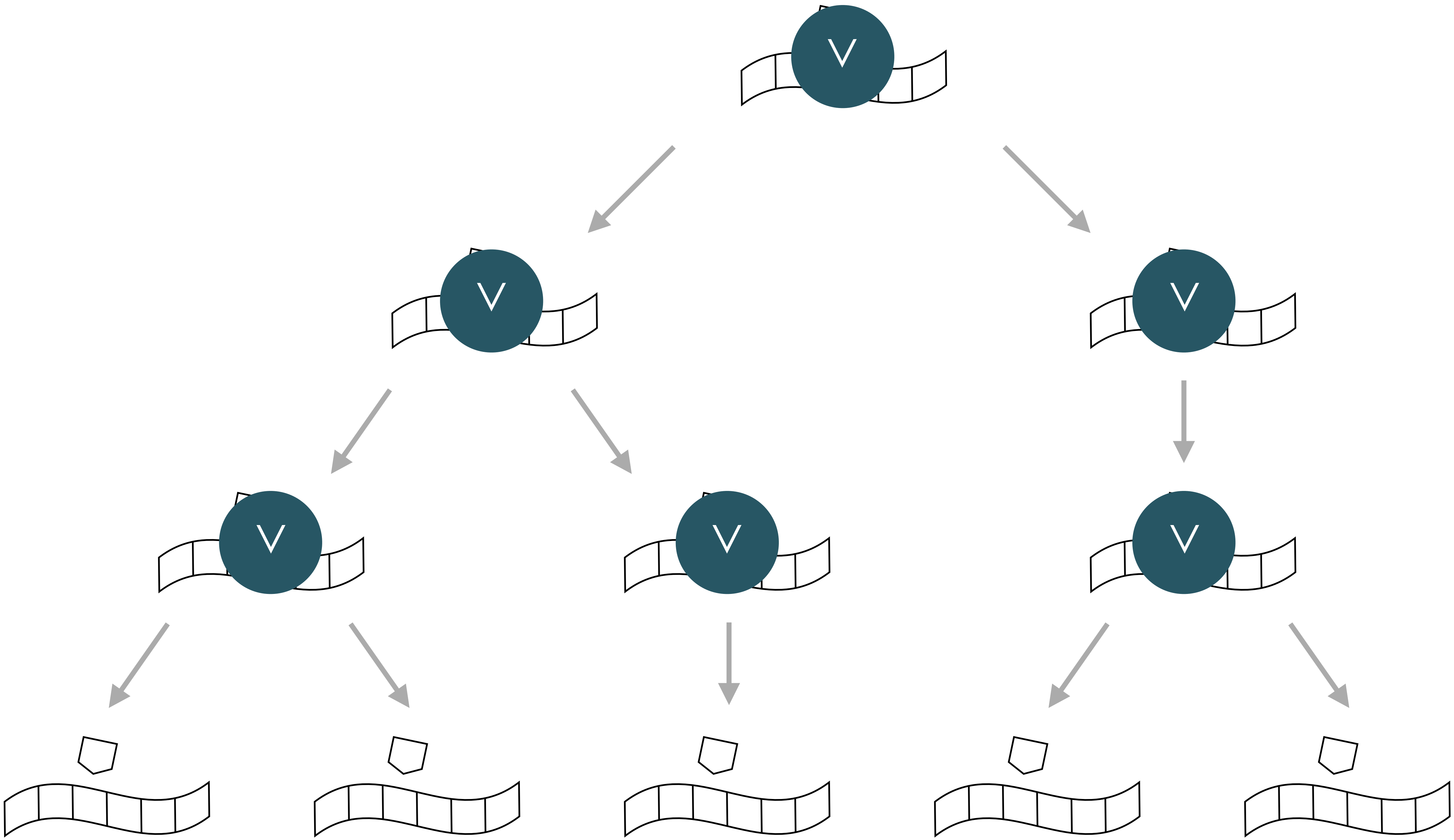
The second machine class and **PSPACE**

- Computing models that solve in polynomial **time** what a Turing machine solves in polynomial **space**
  - Alternating Turing machines
  - Random access machines with arithmetic operations + − x ÷
  - Parallel processes generated by `fork(2)` running on an unbounded number of processors
  - Cellular automata over hyperbolic grids

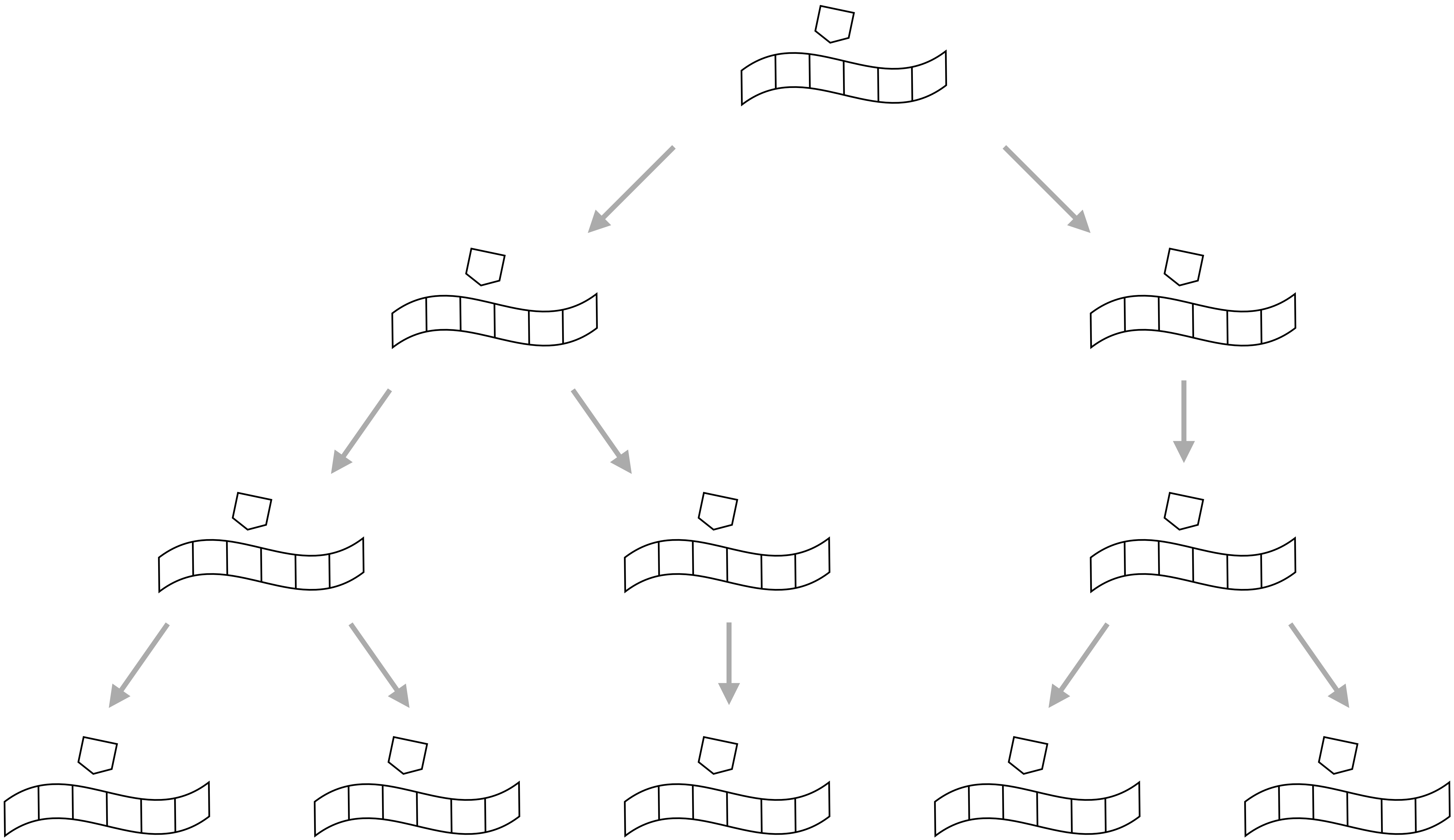
Nondeterministic Turing machines: **NP**



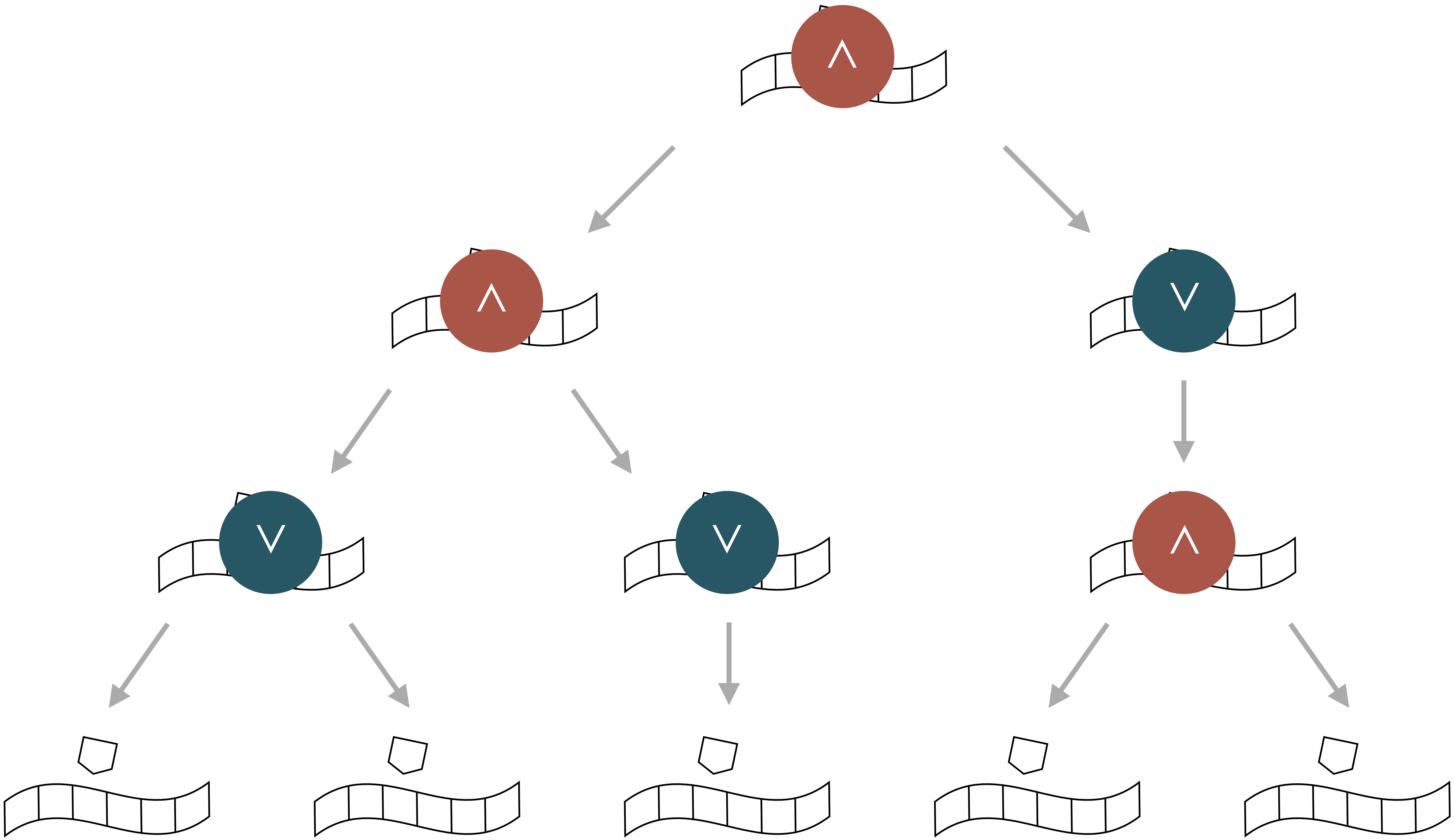
Nondeterministic Turing machines: **NP**



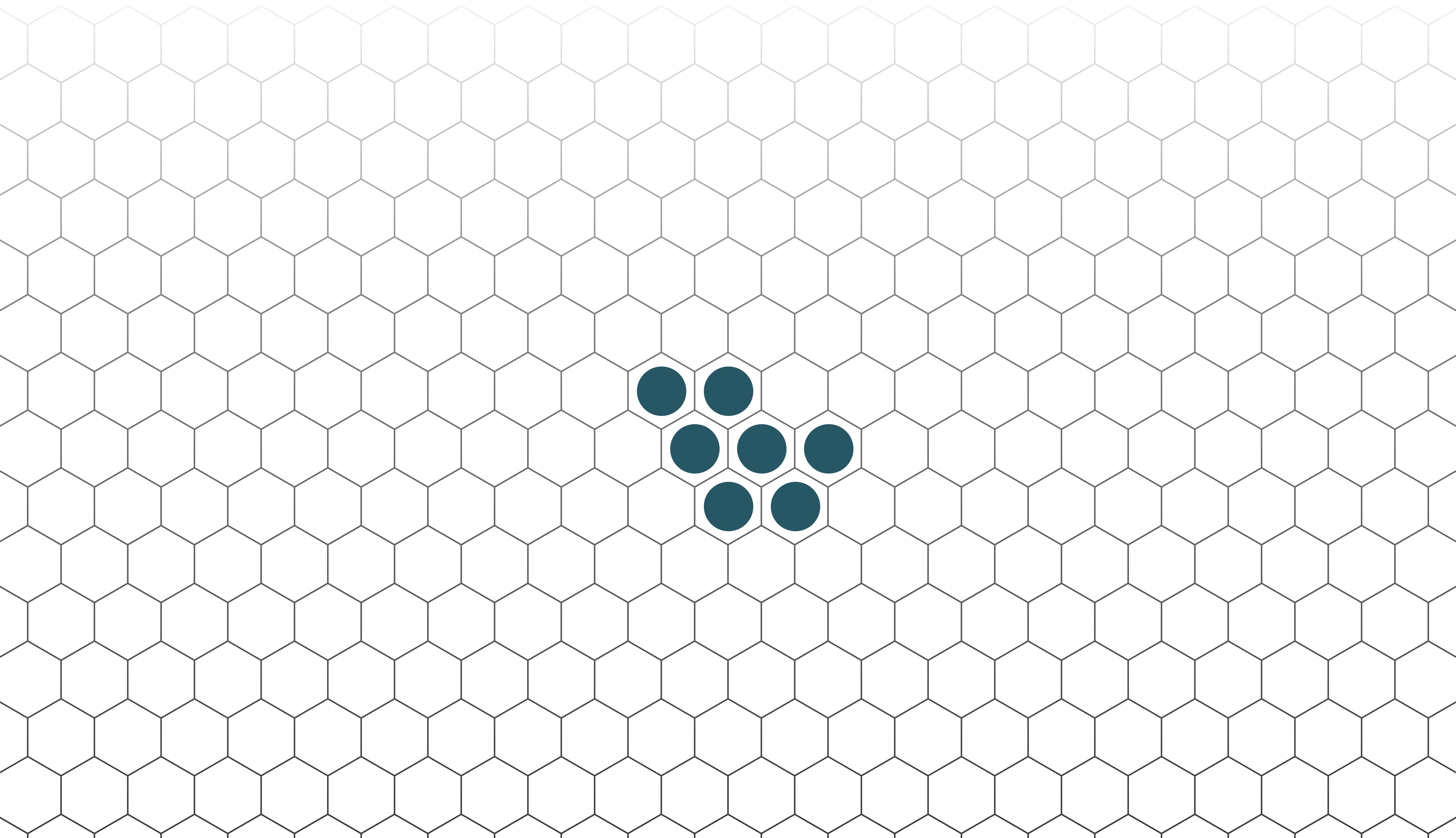
Alternating Turing machines: **PSPACE**



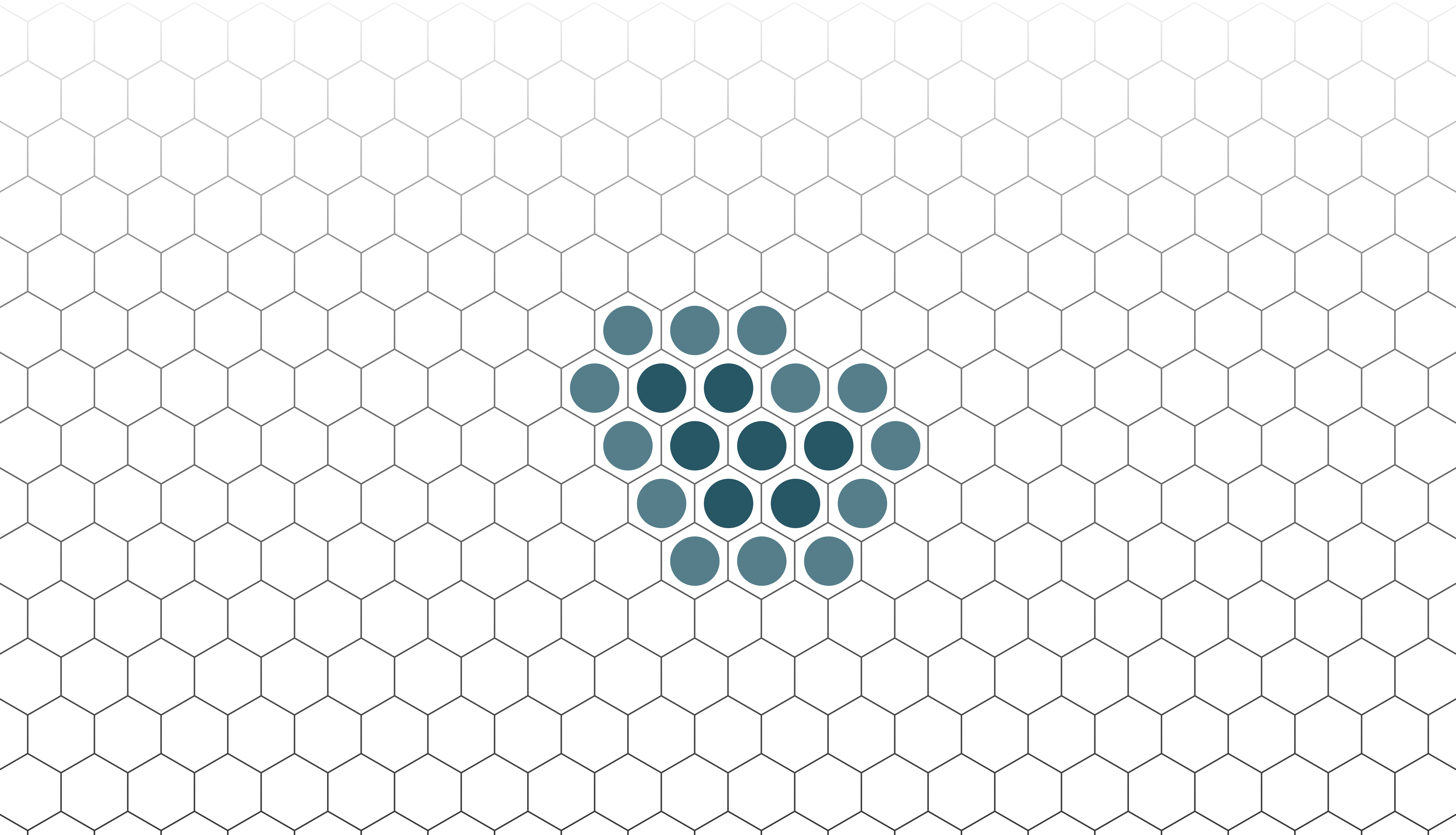
Alternating Turing machines: **PSPACE**



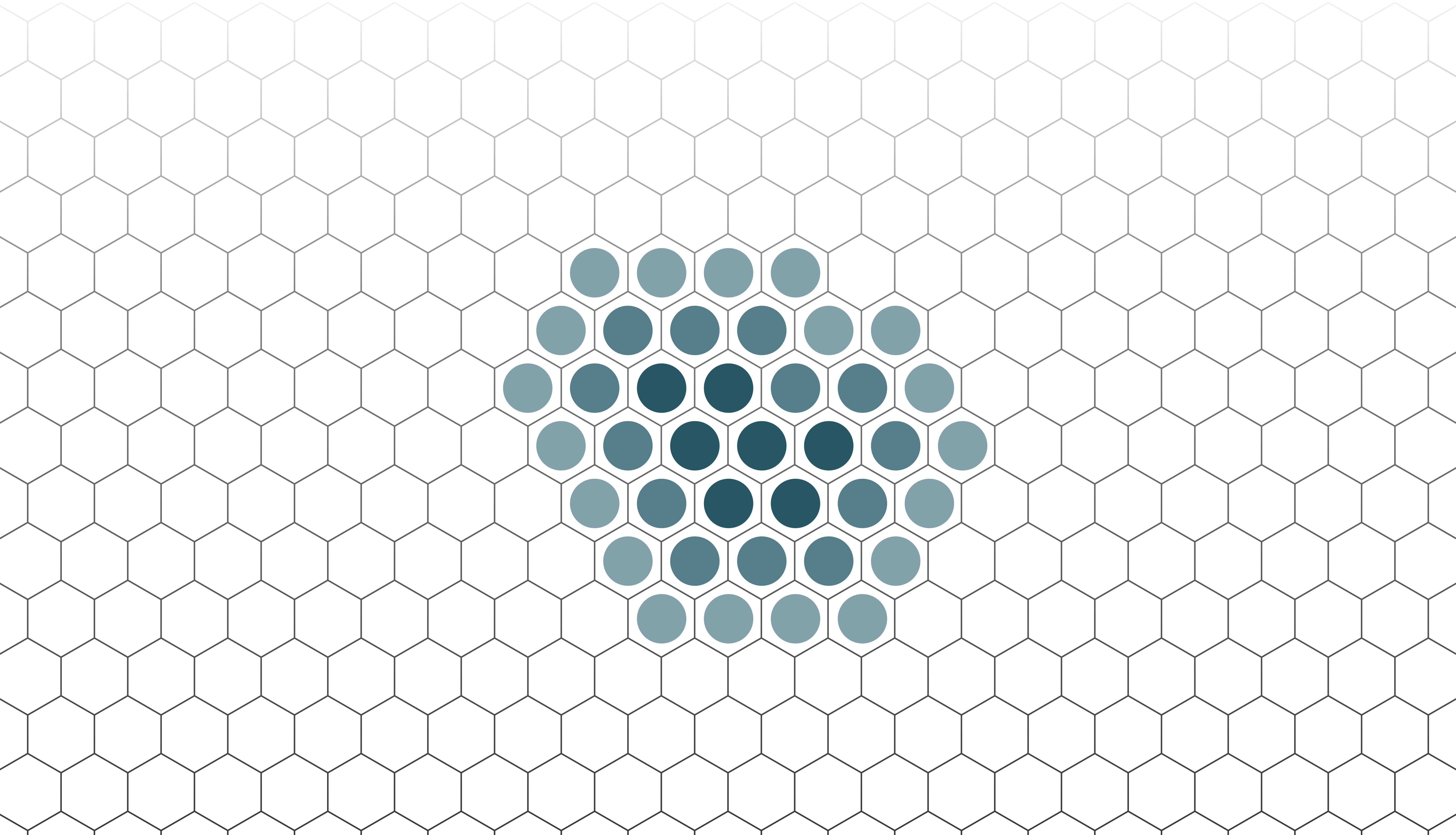
Cellular automata over a Euclidean grid



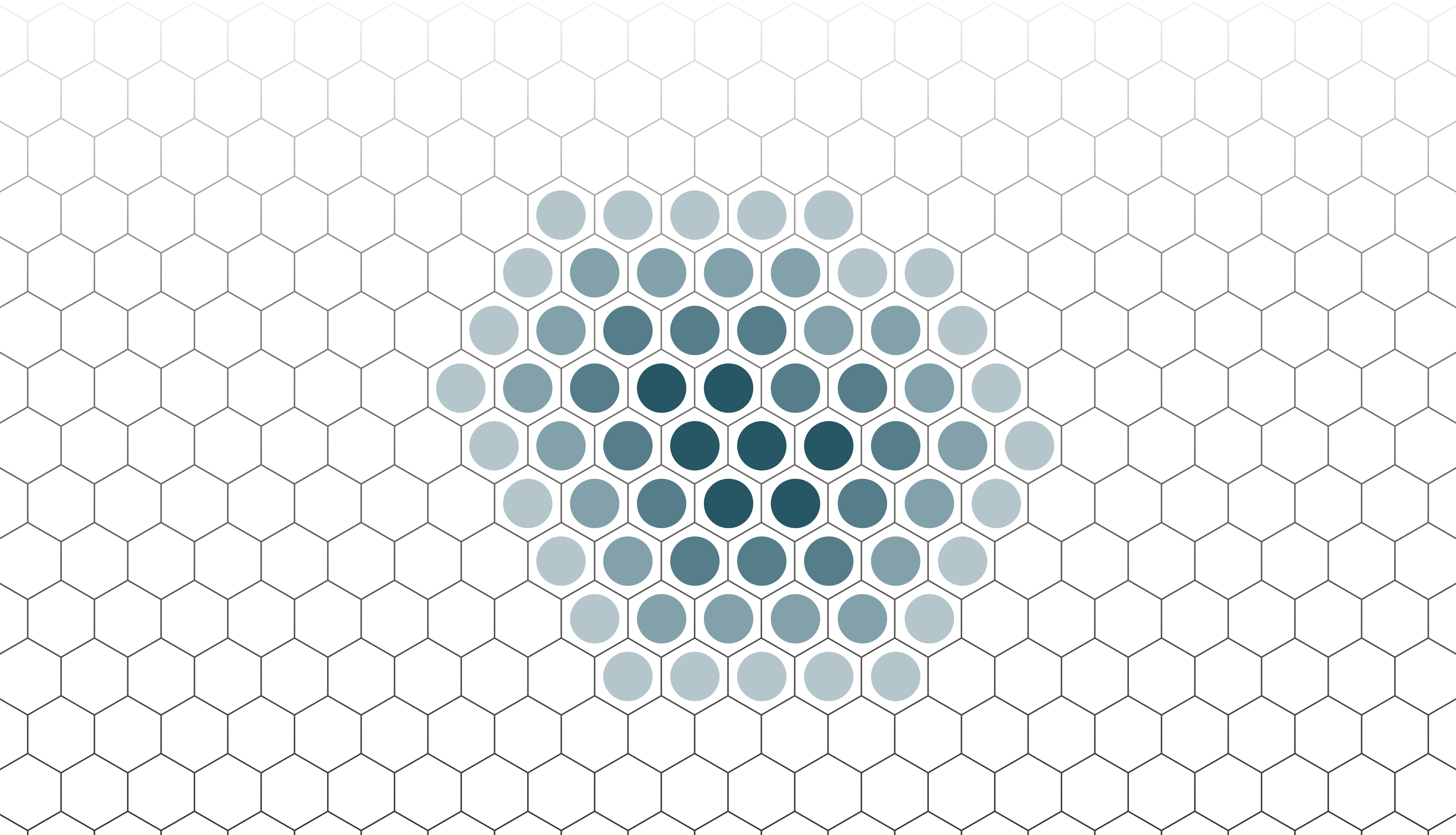
Cellular automata over a Euclidean grid



Cellular automata over a Euclidean grid

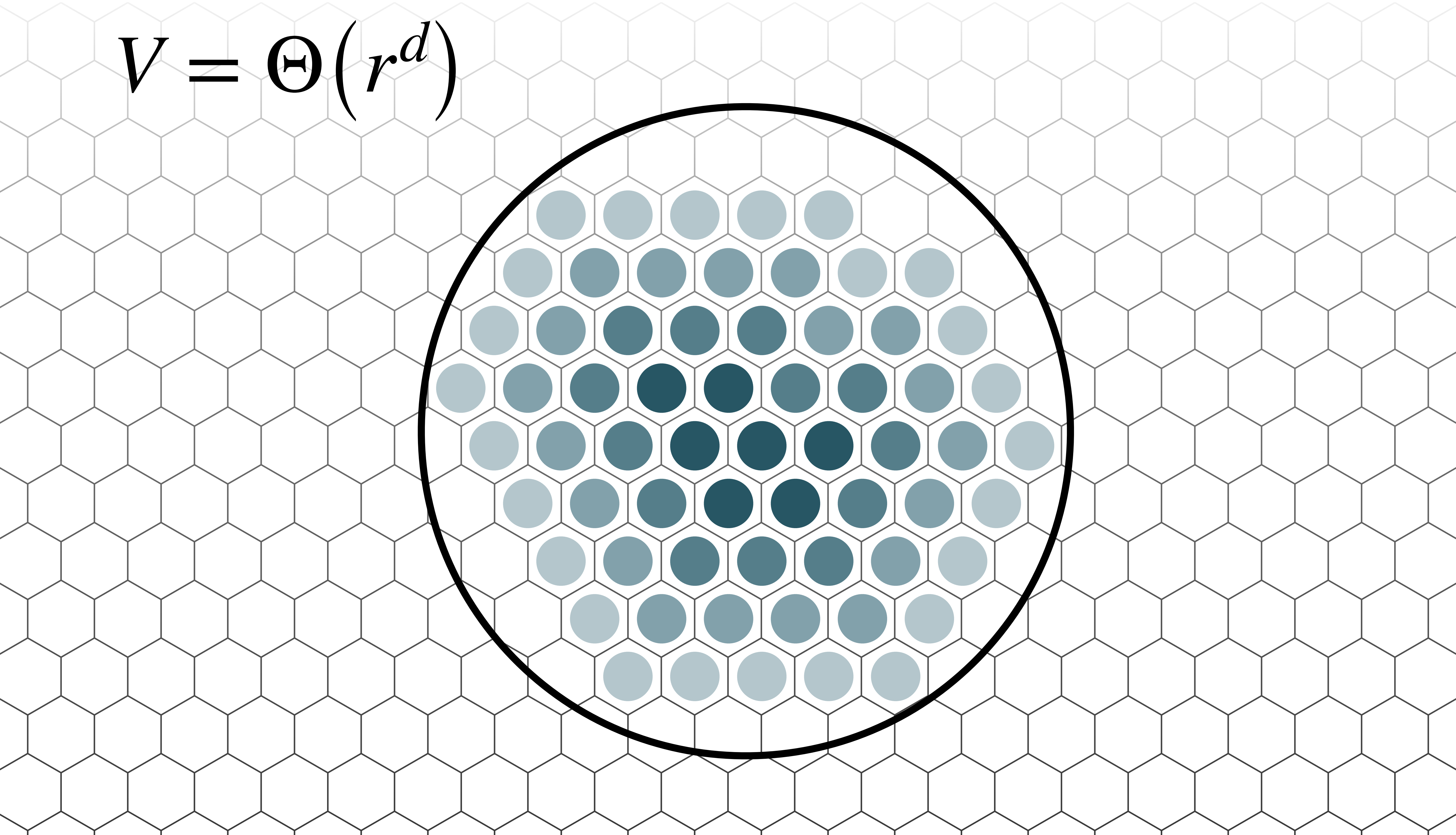


Cellular automata over a Euclidean grid

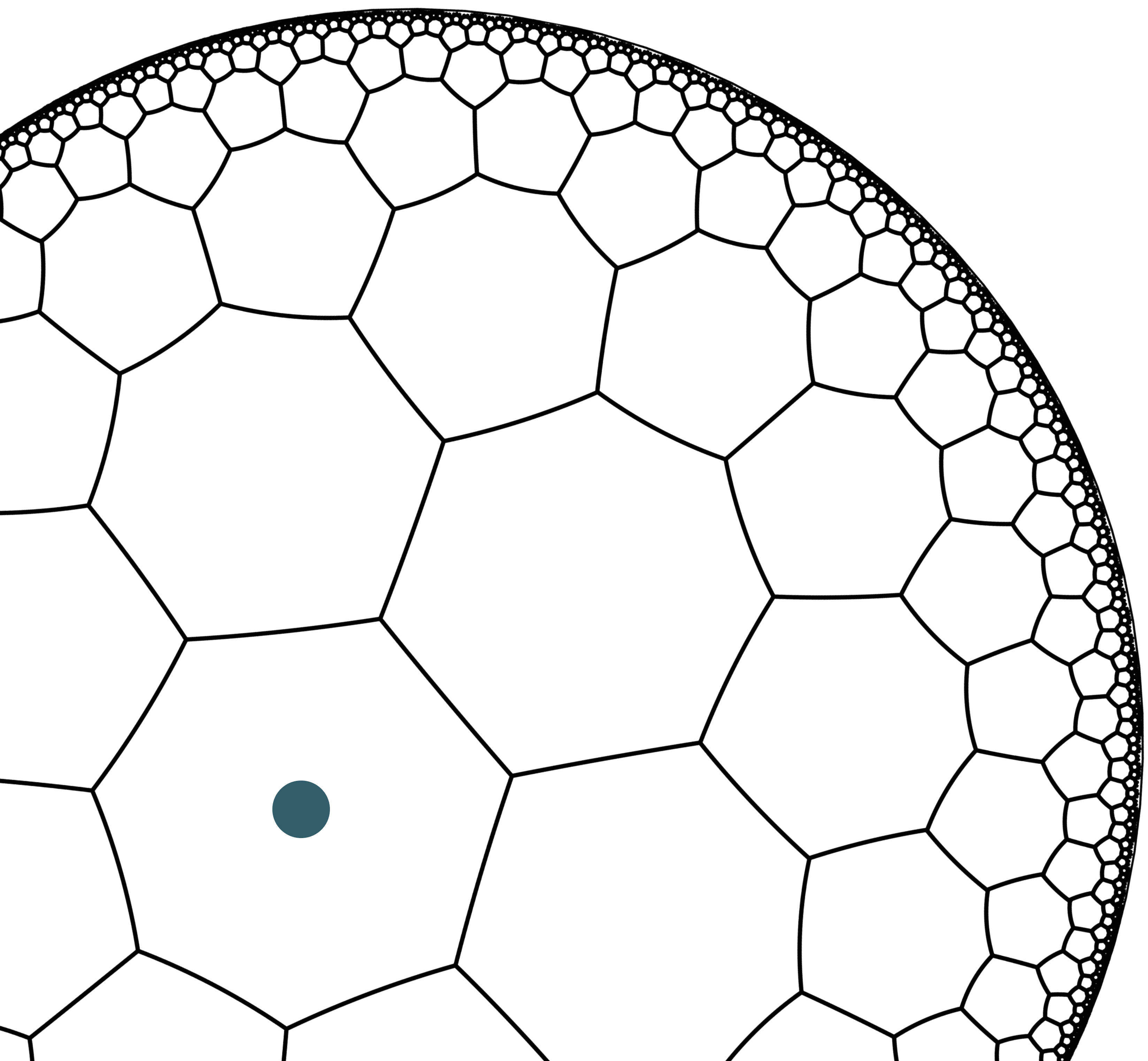


Cellular automata over a Euclidean grid

$$V = \Theta\left(r^d\right)$$

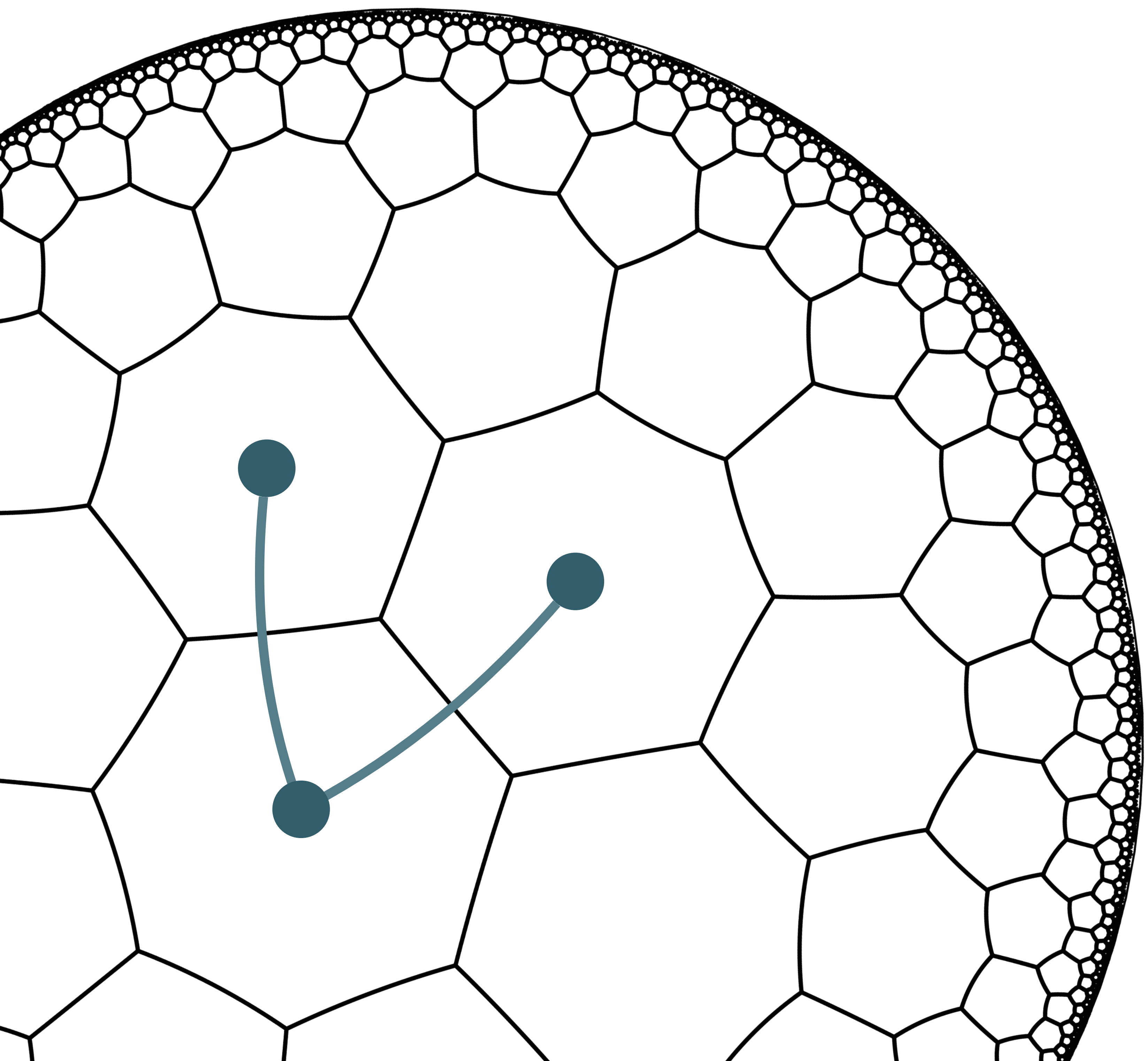


Hyperbolic cellular automata



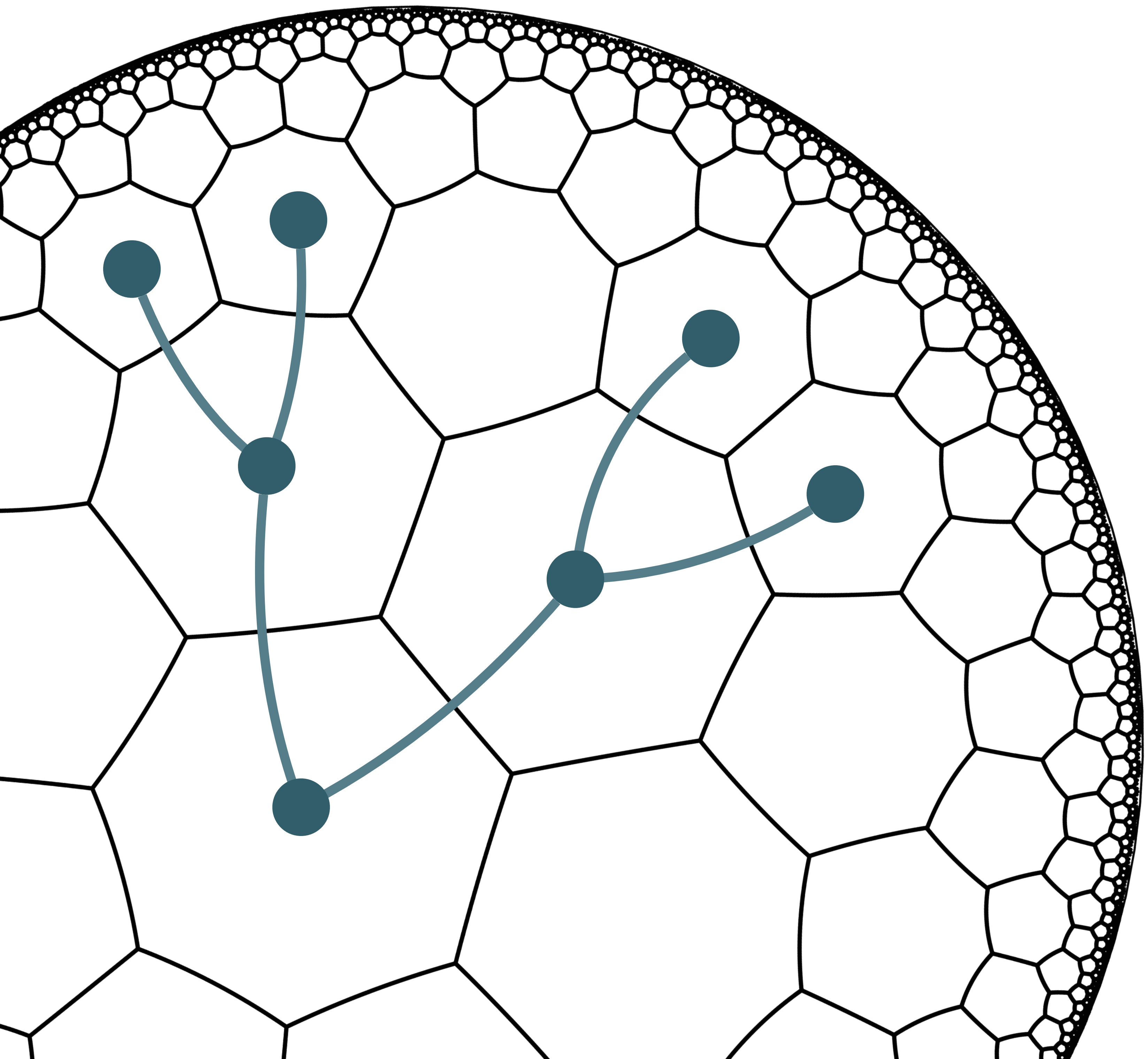
Pavage du plan hyperbolique par des heptagones, dans le modèle du disque de Poincaré. By Theon, used under CC BY-SA 3.0 https:/en.wikipedia.org/wiki/File:PavageHypPoincare2.svg

Hyperbolic cellular automata



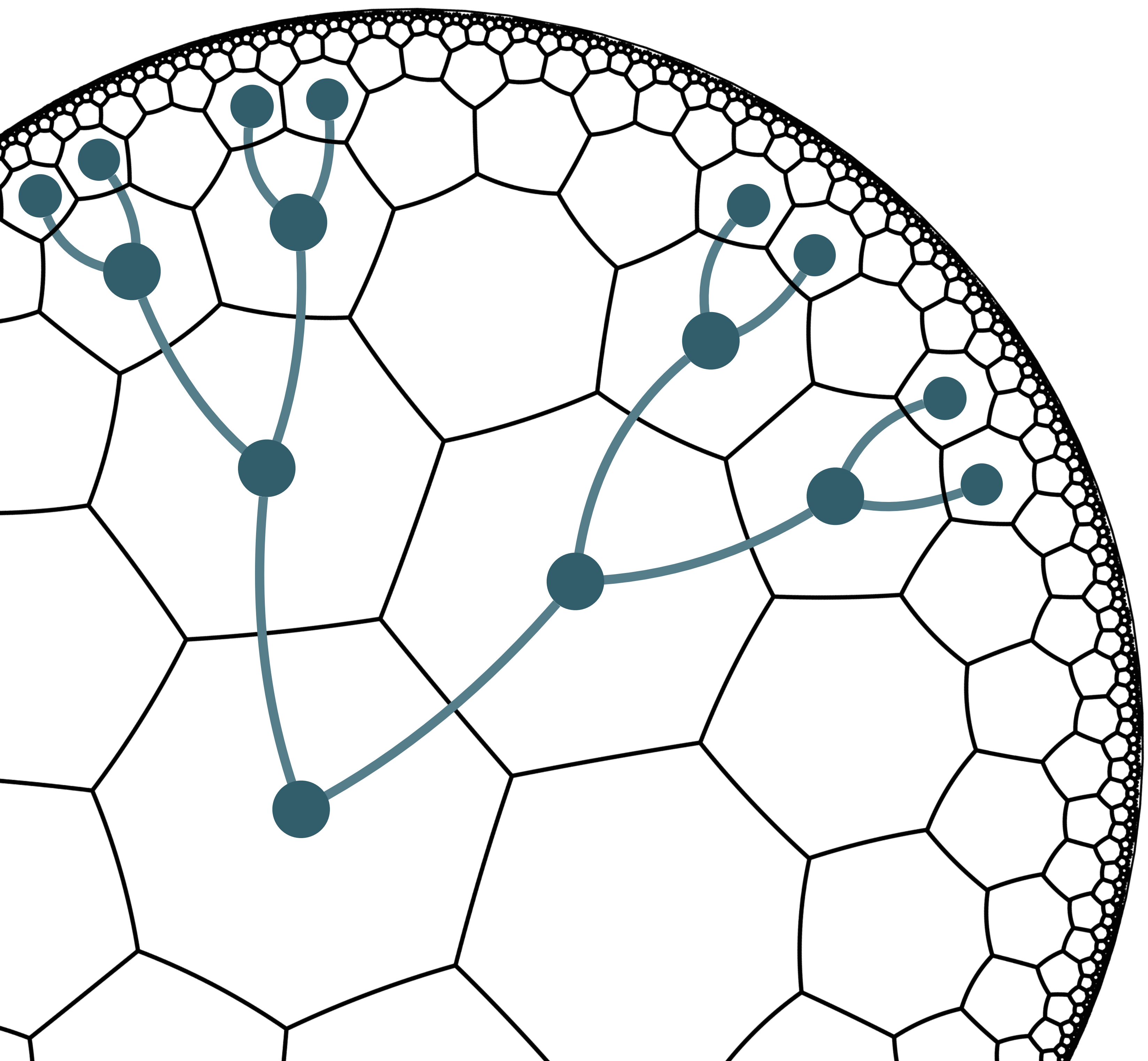
Pavage du plan hyperbolique par des heptagones, dans le modèle du disque de Poincaré. By Theon, used under CC BY-SA 3.0 https:/en.wikipedia.org/wiki/File:PavageHypPoincare2.svg

Hyperbolic cellular automata



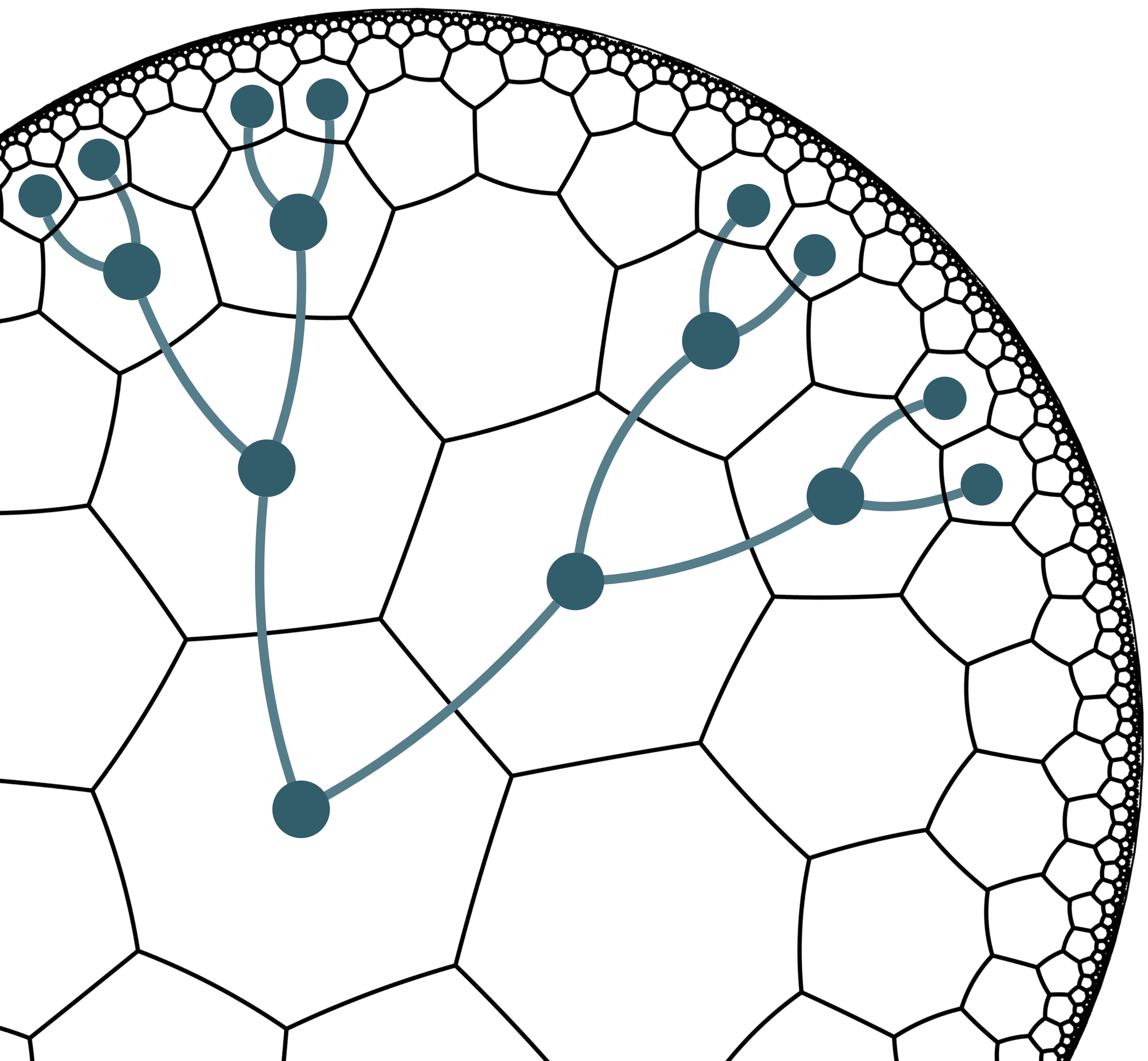
Pavage du plan hyperbolique par des heptagones, dans le modèle du disque de Poincaré. By Theon, used under CC BY-SA 3.0 https:/en.wikipedia.org/wiki/File:PavageHypPoincare2.svg

Hyperbolic cellular automata



Pavage du plan hyperbolique par des heptagones, dans le modèle du disque de Poincaré. By Theon, used under CC BY-SA 3.0 https:/en.wikipedia.org/wiki/File:PavageHypPoincare2.svg

Hyperbolic cellular automata



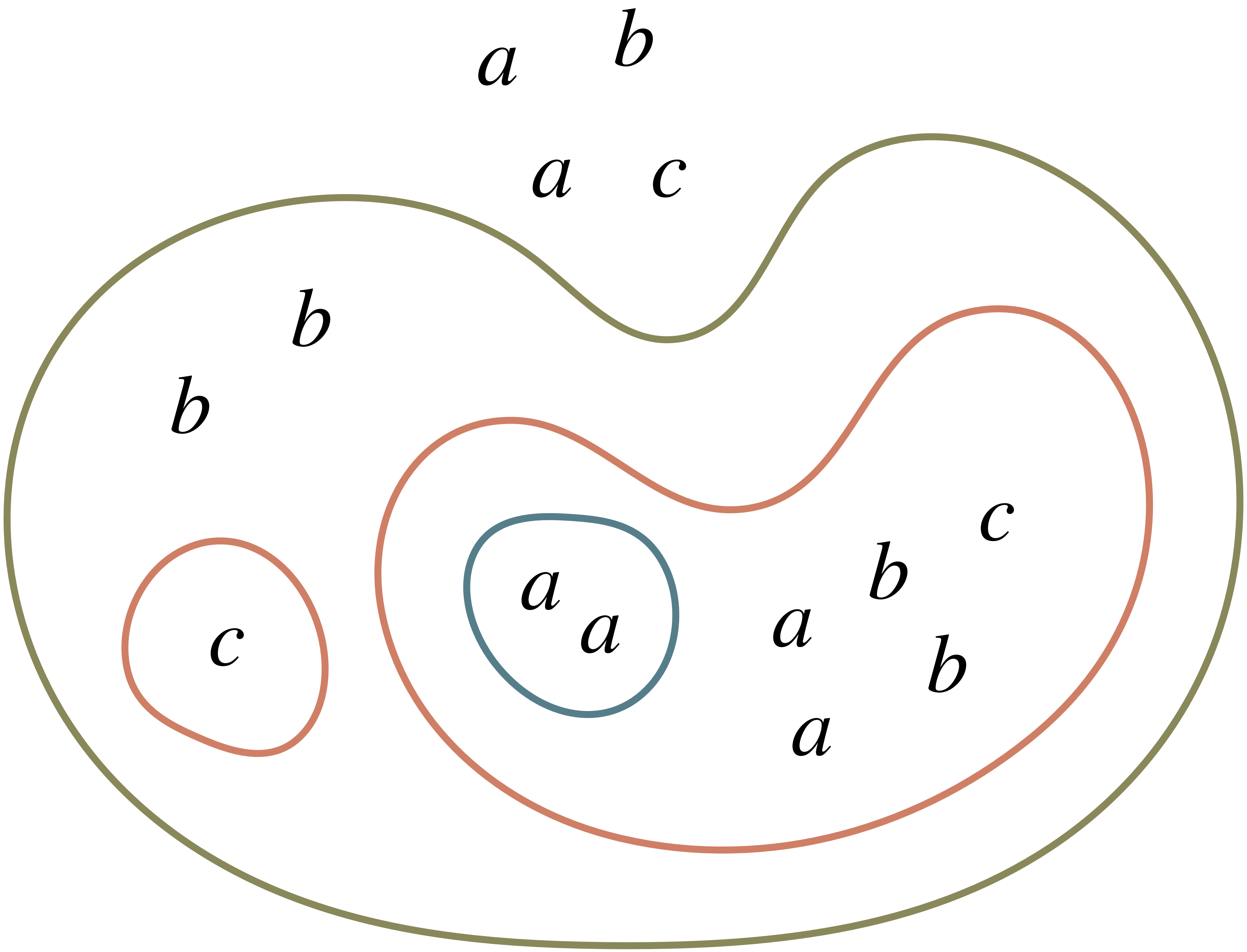
$$V = \Omega\left(2^r\right)$$

Pavage du plan hyperbolique par des heptagones, dans le modèle du disque de Poincaré. By Theon, used under CC BY-SA 3.0 https:/en.wikipedia.org/wiki/File:PavageHypPoincare2.svg

Rule of thumb

- Sequential machines are first class
- Bounded parallel machines are first class
- Unbounded parallel machines are second class

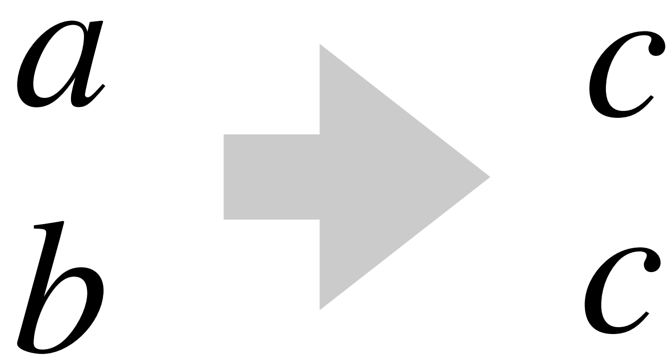
Membrane systems (P systems)

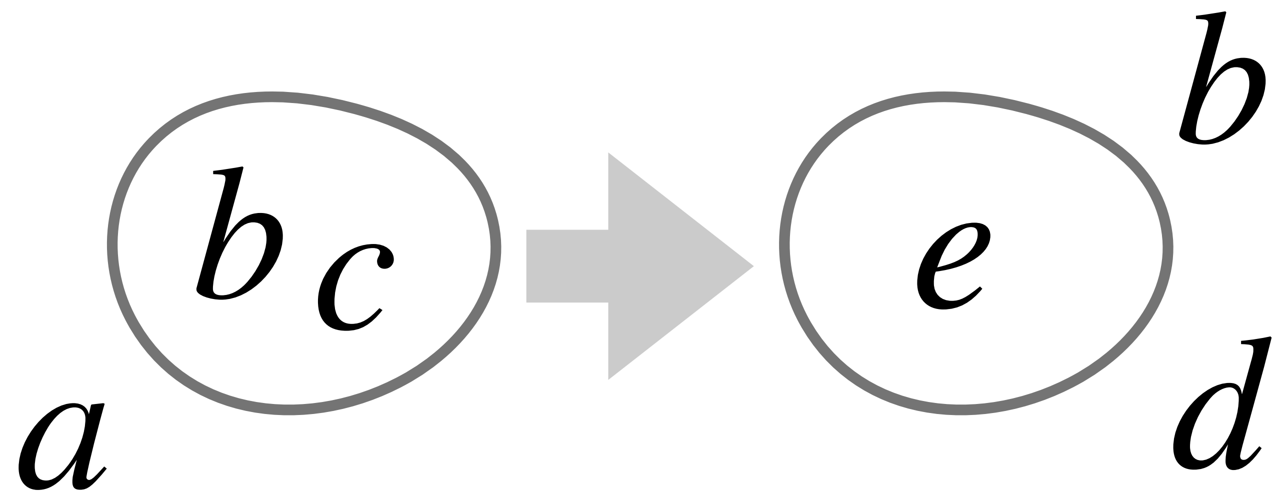


Evolution rules and maximally parallel semantics

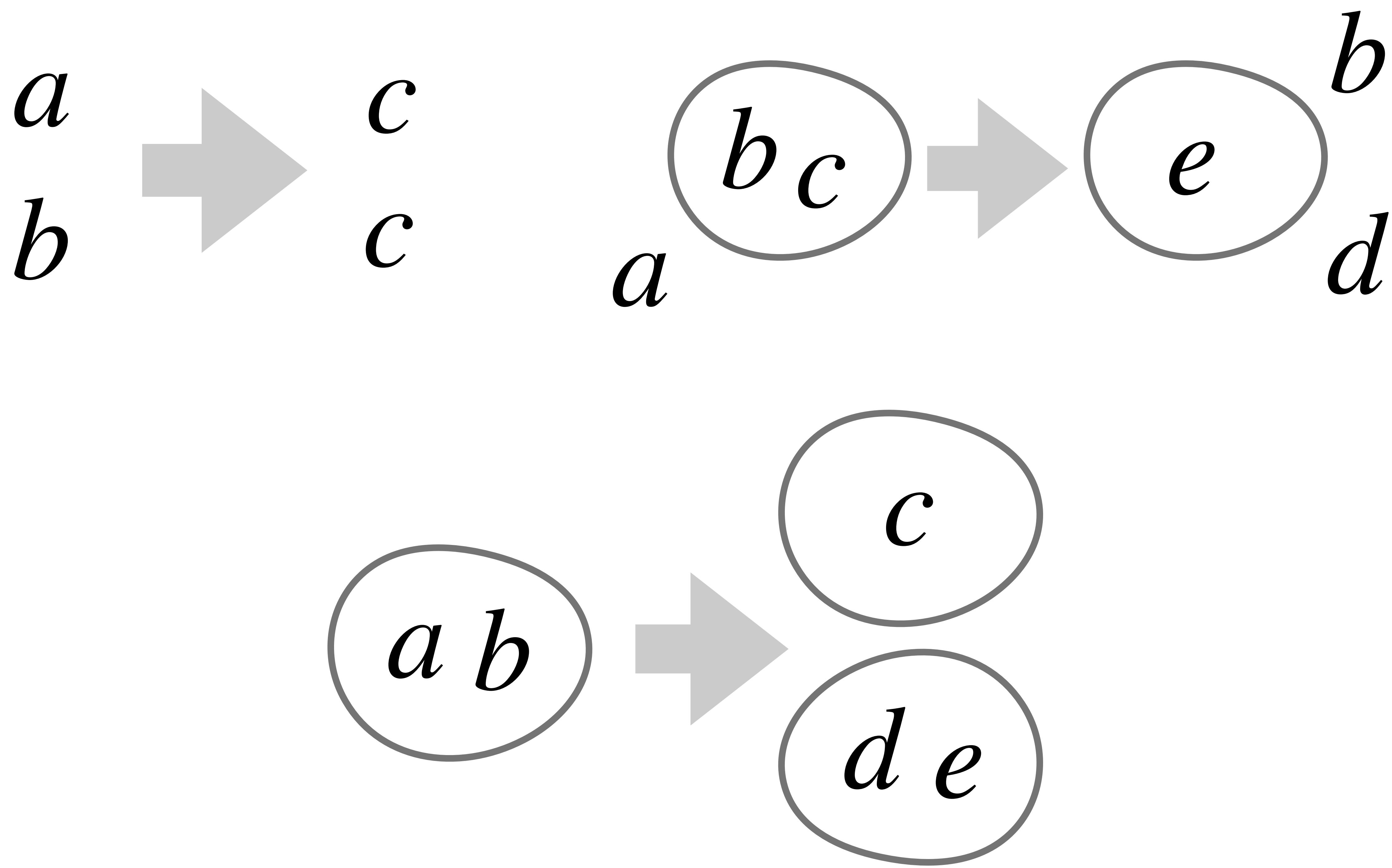
$$\begin{matrix} a \\ b \end{matrix} \;\Rightarrow\; \begin{matrix} c \\ c \end{matrix}$$

Evolution rules and maximally parallel semantics





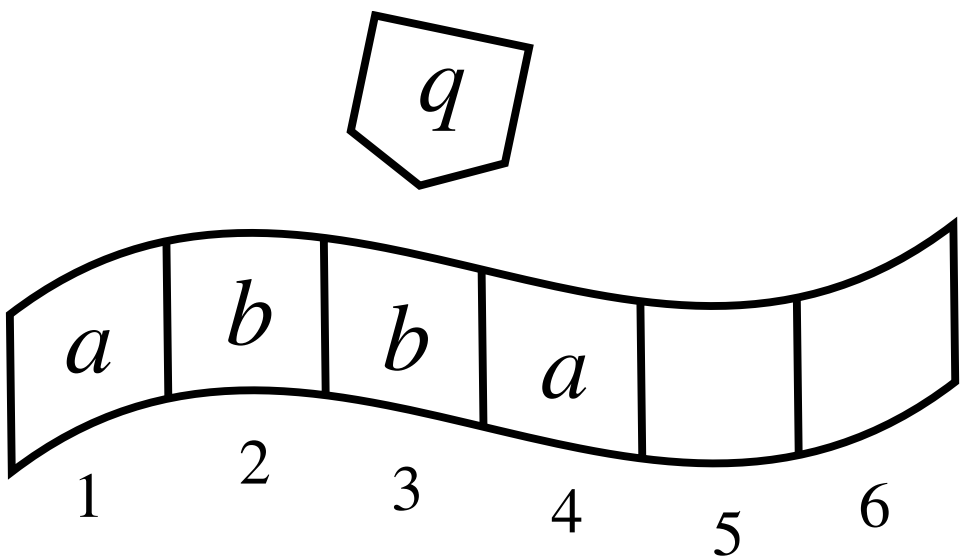
Evolution rules and maximally parallel semantics



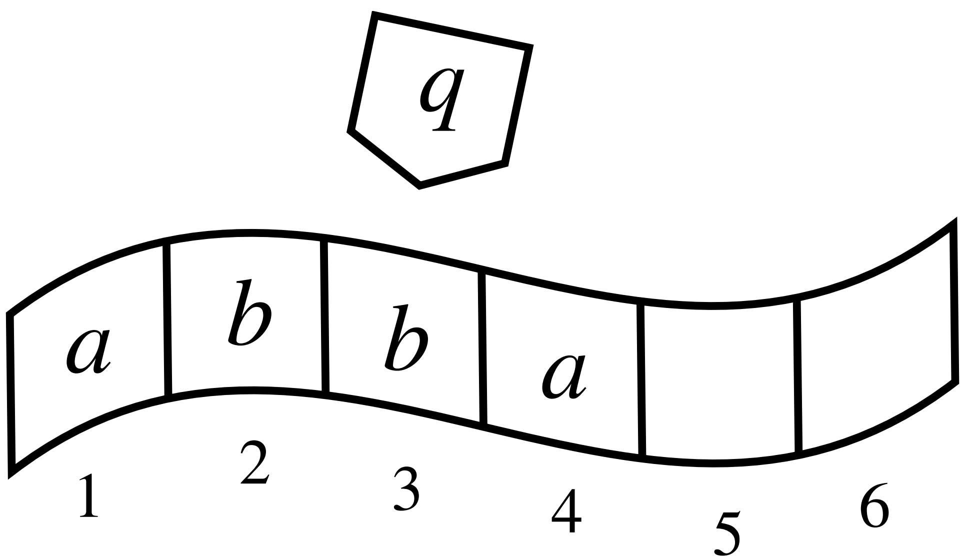
Computational universality of membrane systems

- Able to simulate, e.g., counter machines
  when working in maximally parallel way
- Equivalent to Petri nets or vector addition systems
  when working in an asynchronous mode
- Which means that they are not computationally universal
- But see W. Czerwinski, S. Lasota, R. Lazic, J. Leroux, F. Mazowiecki, **The Reachability Problem for Petri Nets is Not Elementary**, https://arxiv.org/abs/1809.07115

Simulating bounded-tape Turing machines



Simulating bounded-tape Turing machines



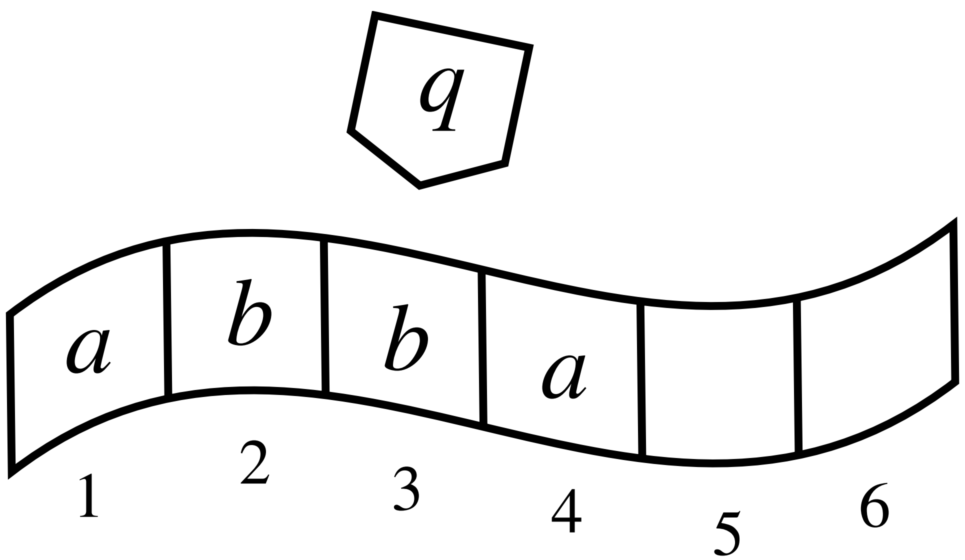
b_2 b_3

a_1

a_4

q_3

Simulating bounded-tape Turing machines



b_2 b_3

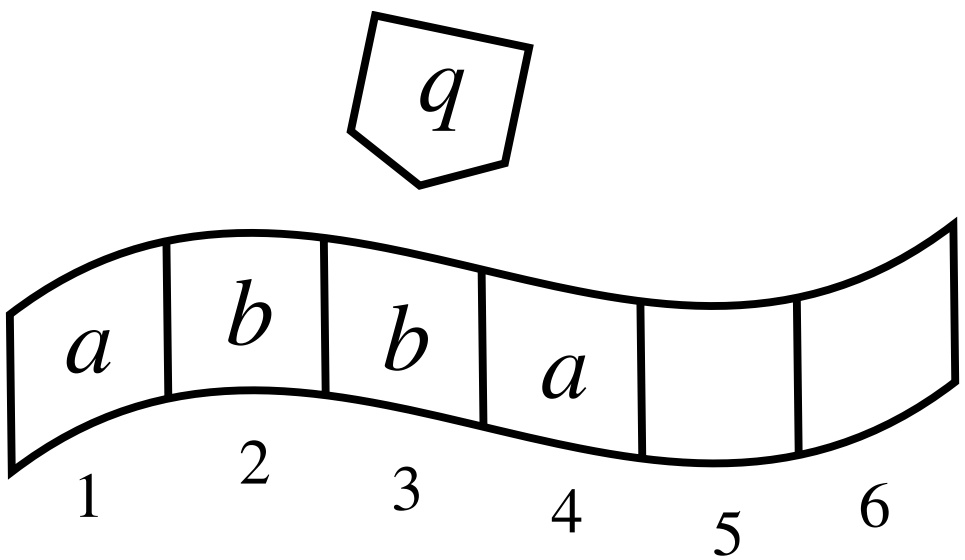
a_1

a_4

q_3

$$\delta(q, b) = (r, a, +1)$$

Simulating bounded-tape Turing machines

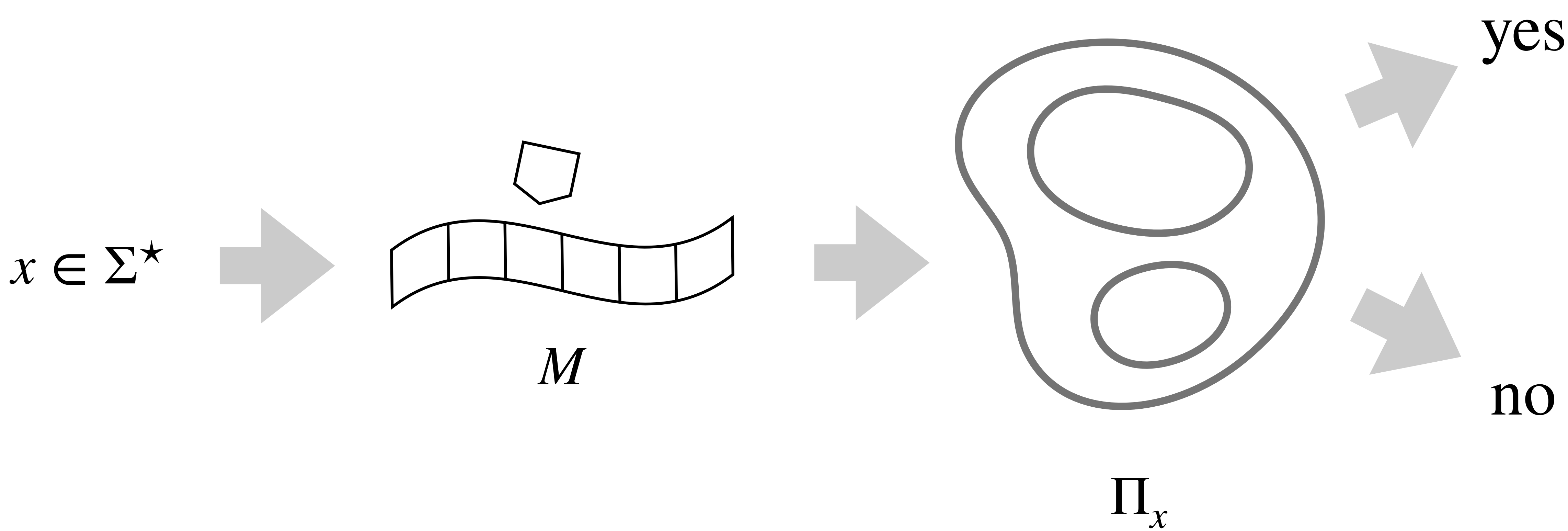


$$b_2 \quad b_3$$
$$a_1 \qquad\qquad a_4$$
$$q_3$$

$$\delta(q, b) = (r, a, +1)$$

$$\begin{matrix} b_i \\ q_i \end{matrix} \Rightarrow \begin{matrix} a_i \\ r_{i+1} \end{matrix}$$

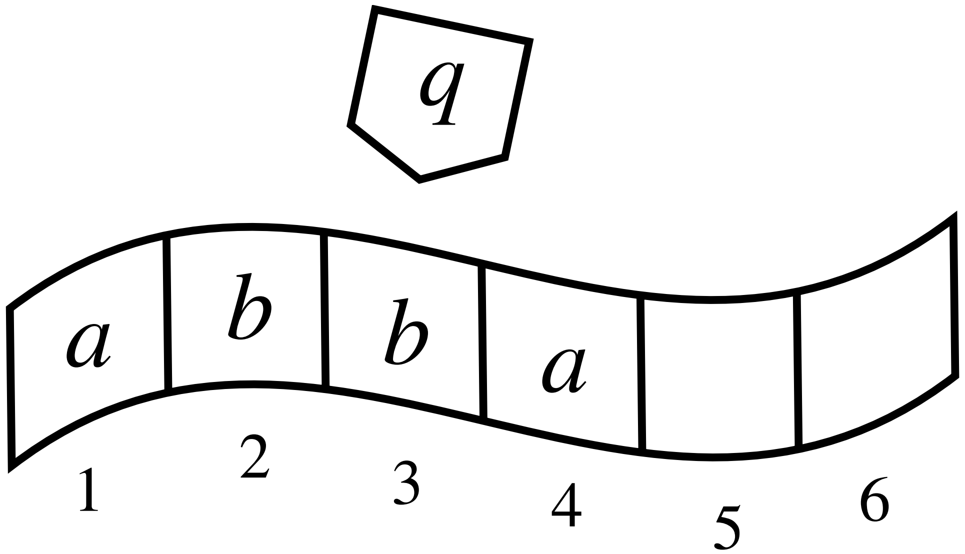
Semi-uniform confluent families of P systems



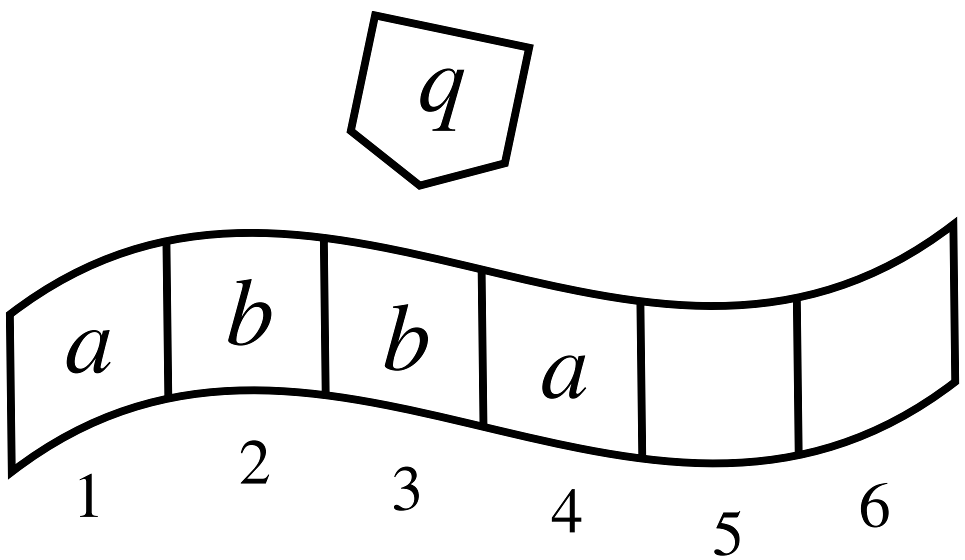
“Milano Theorem”

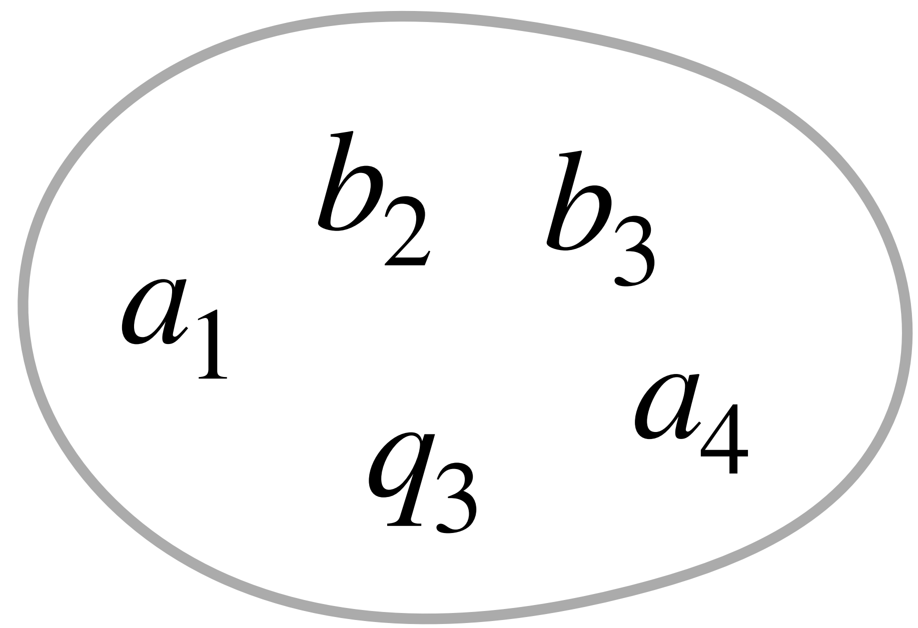
Semi-uniform families of confluent membrane systems **without** membrane division characterise **P** in polynomial time.

Membrane systems solving **NP** problems

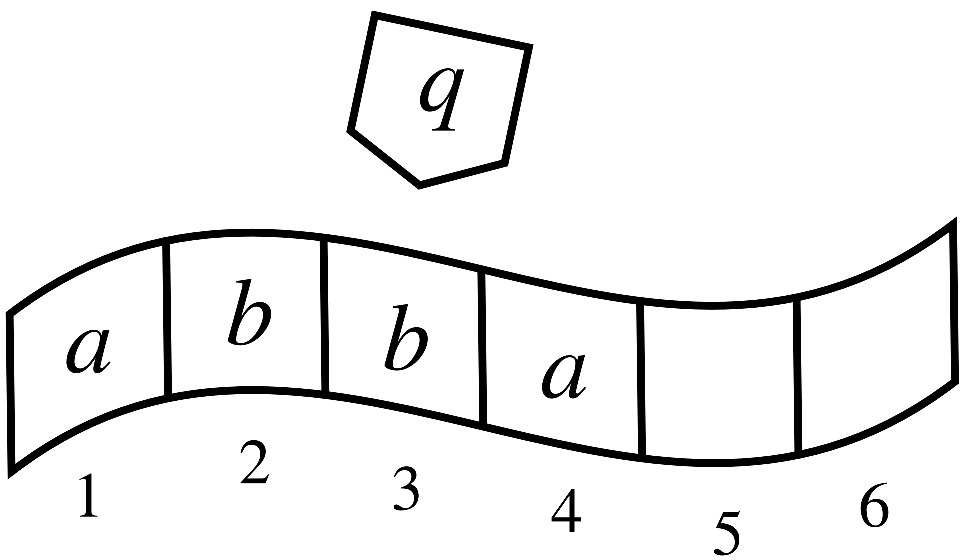


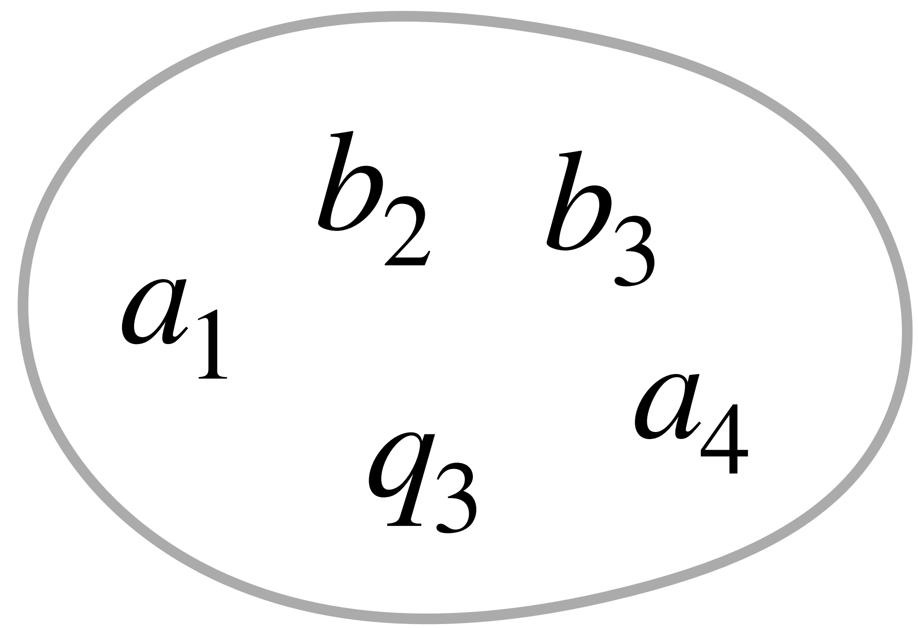
Membrane systems solving **NP** problems





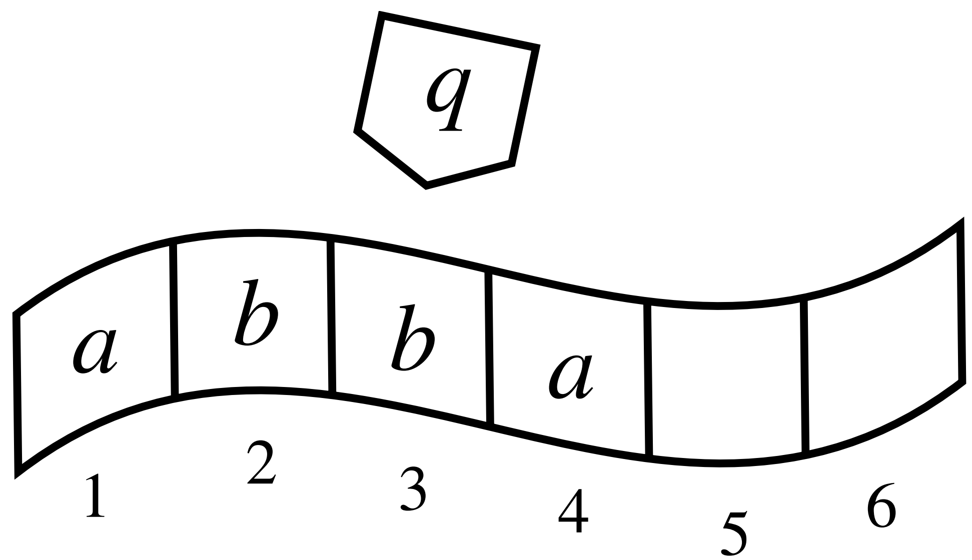
Membrane systems solving **NP** problems

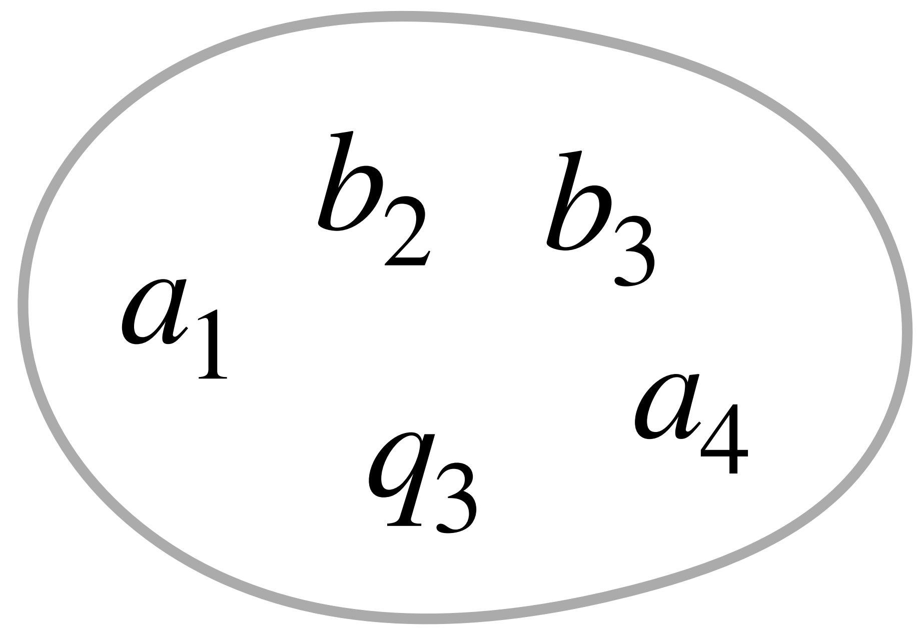




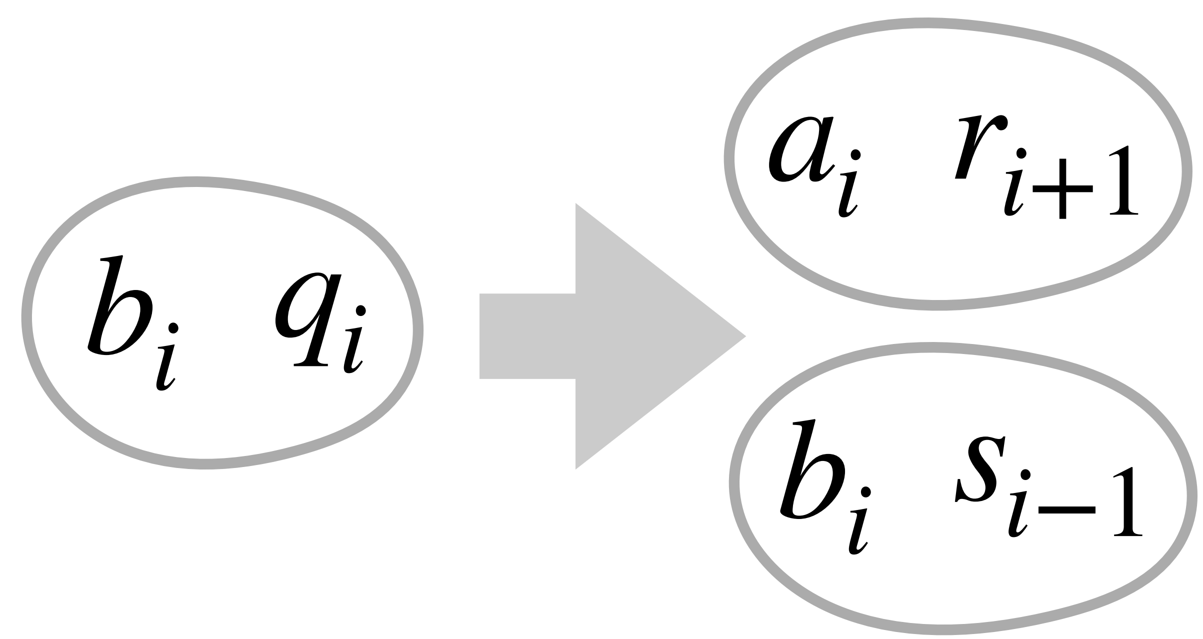
$$\delta(q, b) = \begin{cases} (r, a, +1) \\ (s, b, -1) \end{cases}$$

Membrane systems solving **NP** problems

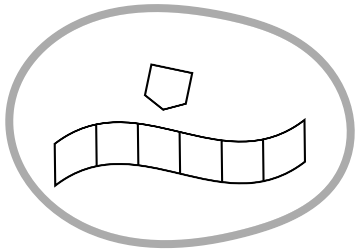




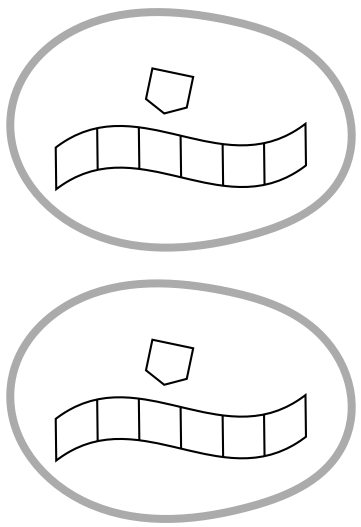
$$\delta(q, b) = \begin{cases} (r, a, +1) \\ (s, b, -1) \end{cases}$$



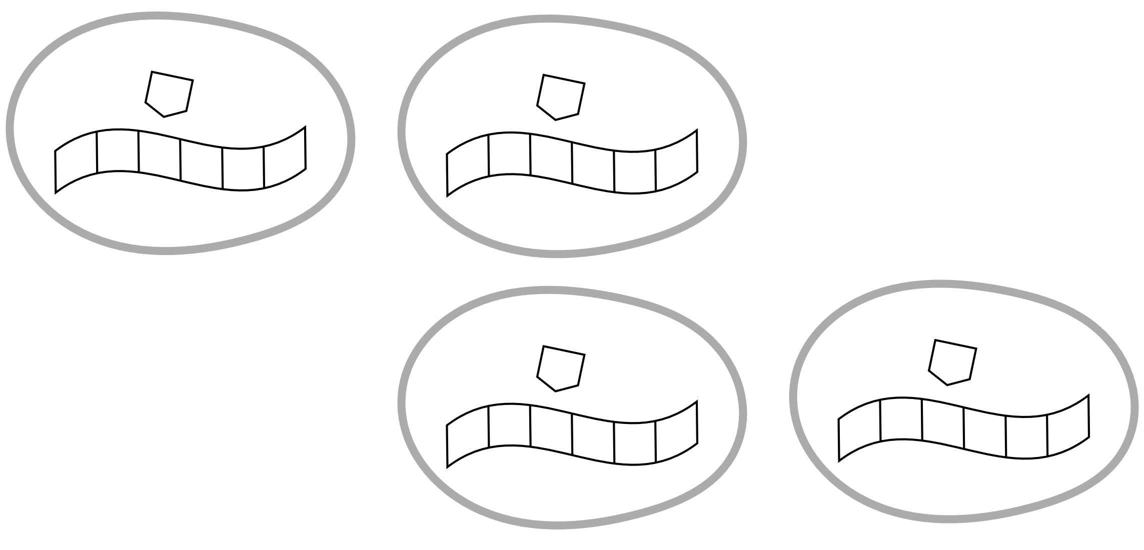
Simulating nondeterminism with parallelism



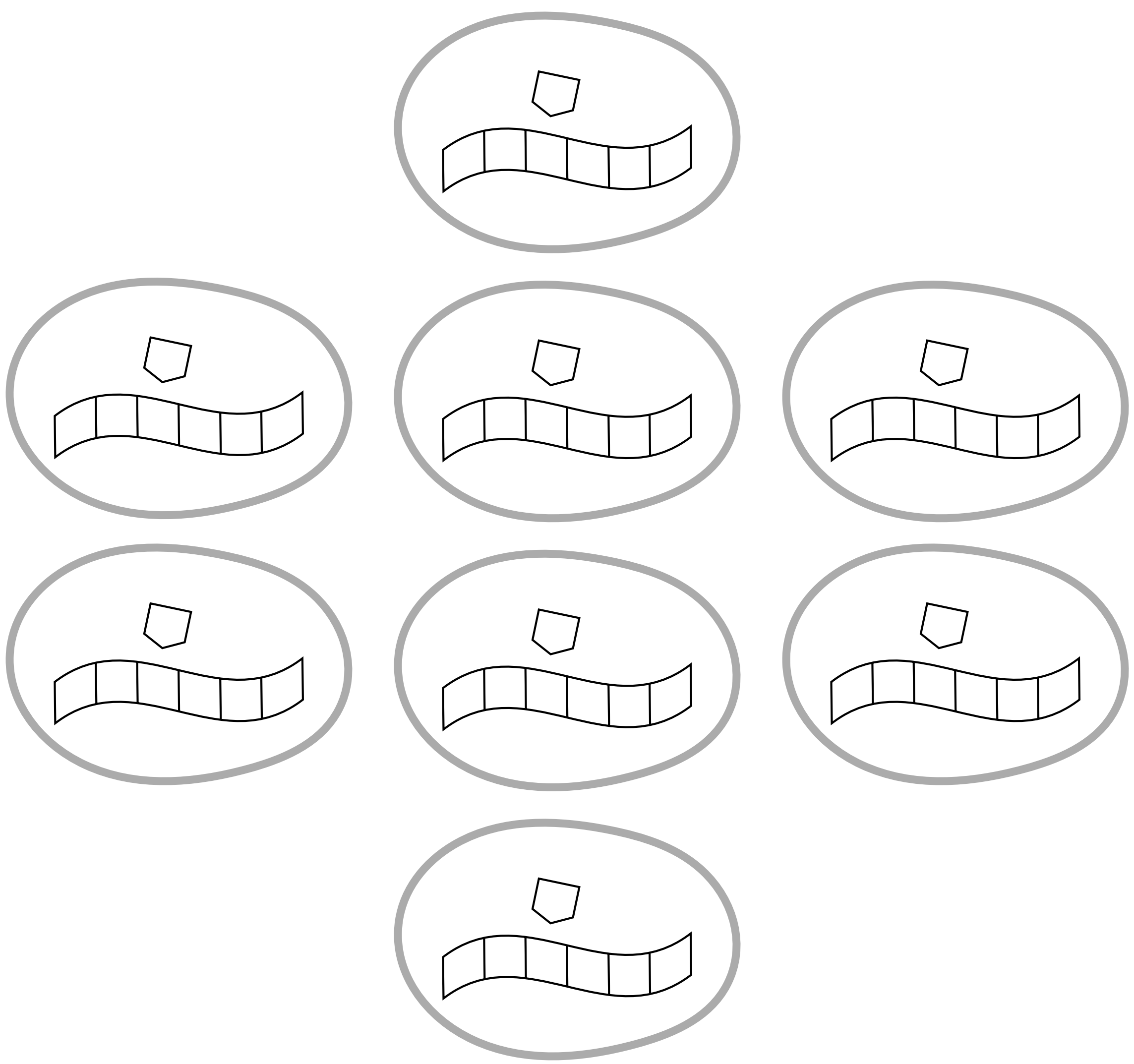
Simulating nondeterminism with parallelism



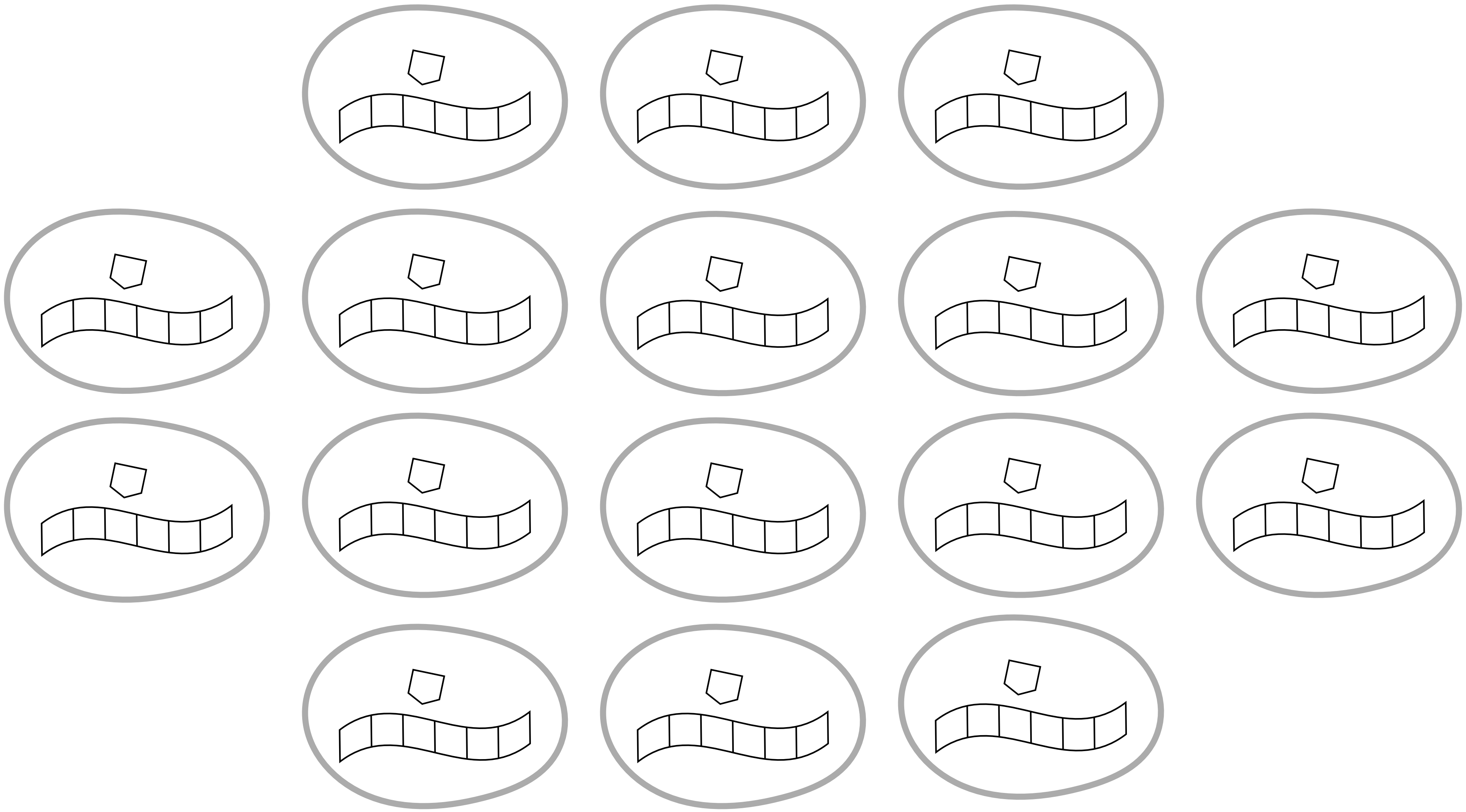
Simulating nondeterminism with parallelism



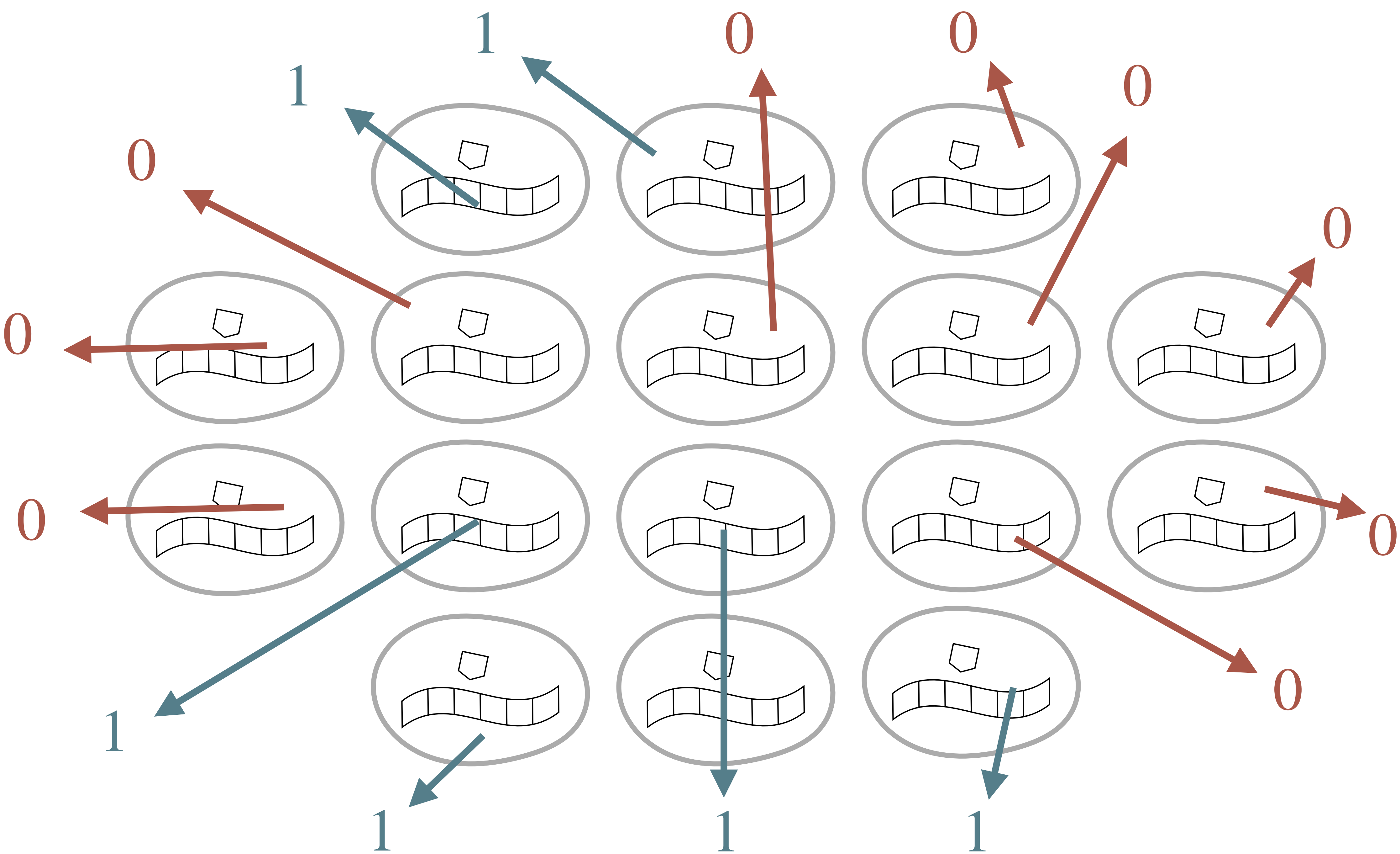
Simulating nondeterminism with parallelism



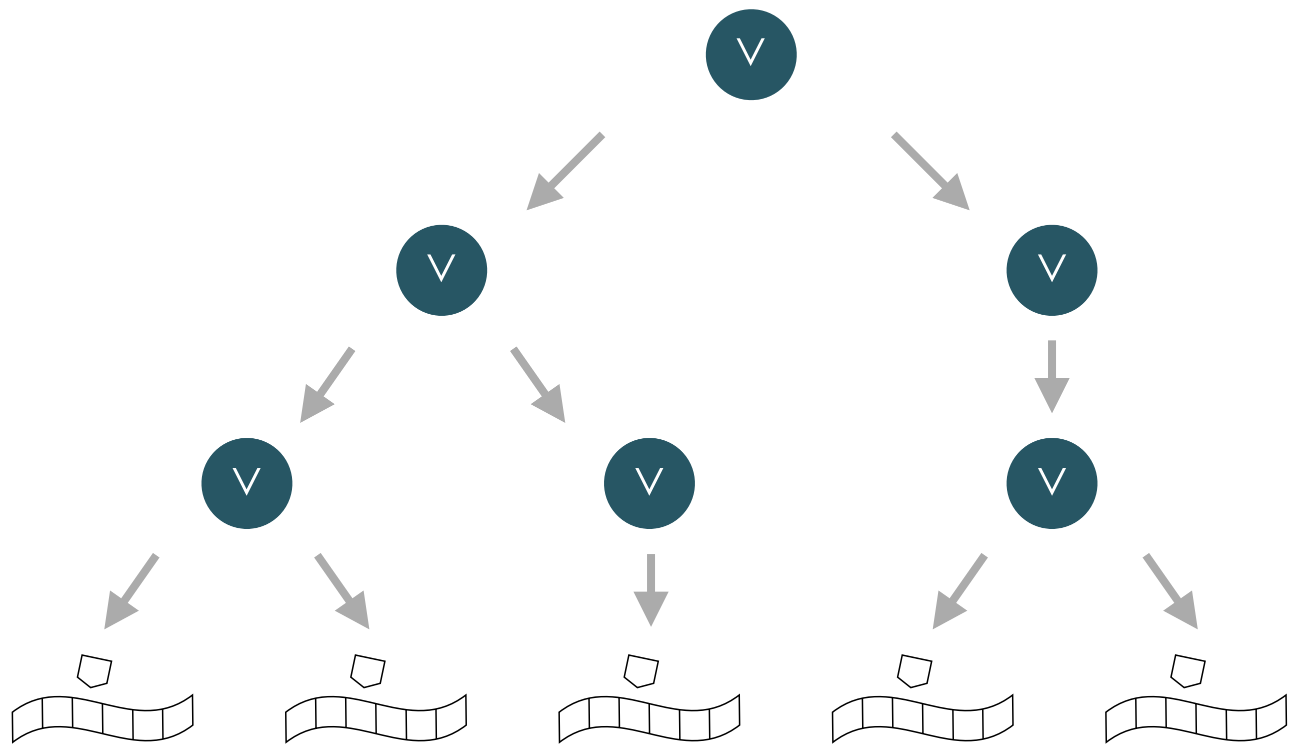
Simulating nondeterminism with parallelism



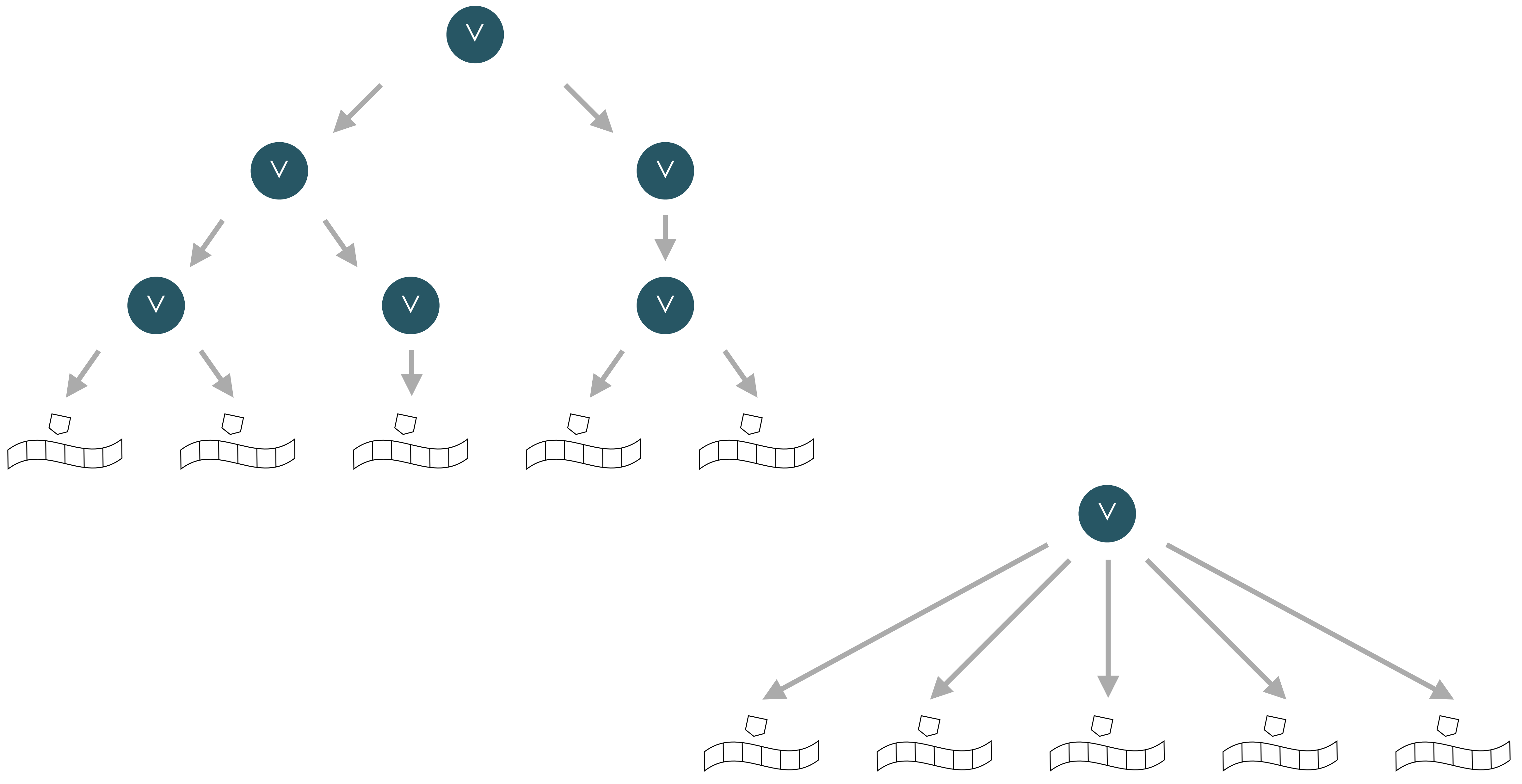
Simulating nondeterminism with parallelism



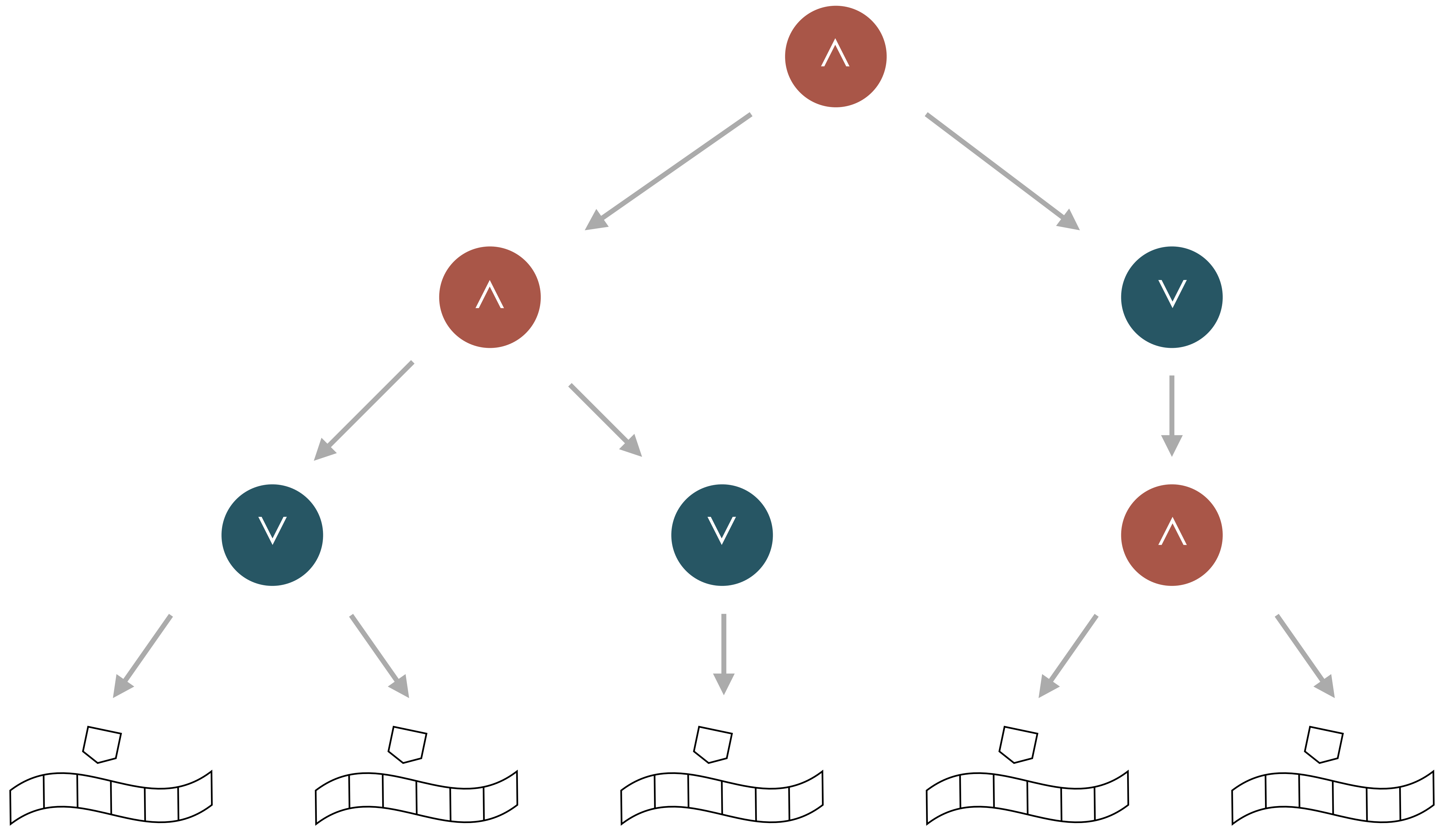
Flattening nondeterministic computation trees



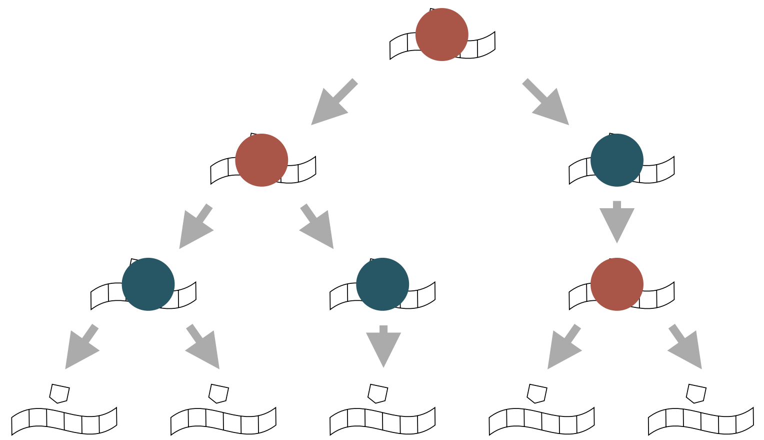
Flattening nondeterministic computation trees



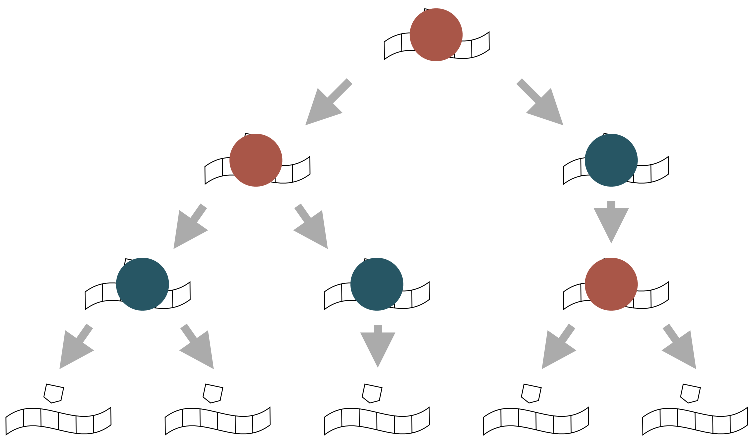
Cannot flatten alternating computation trees!

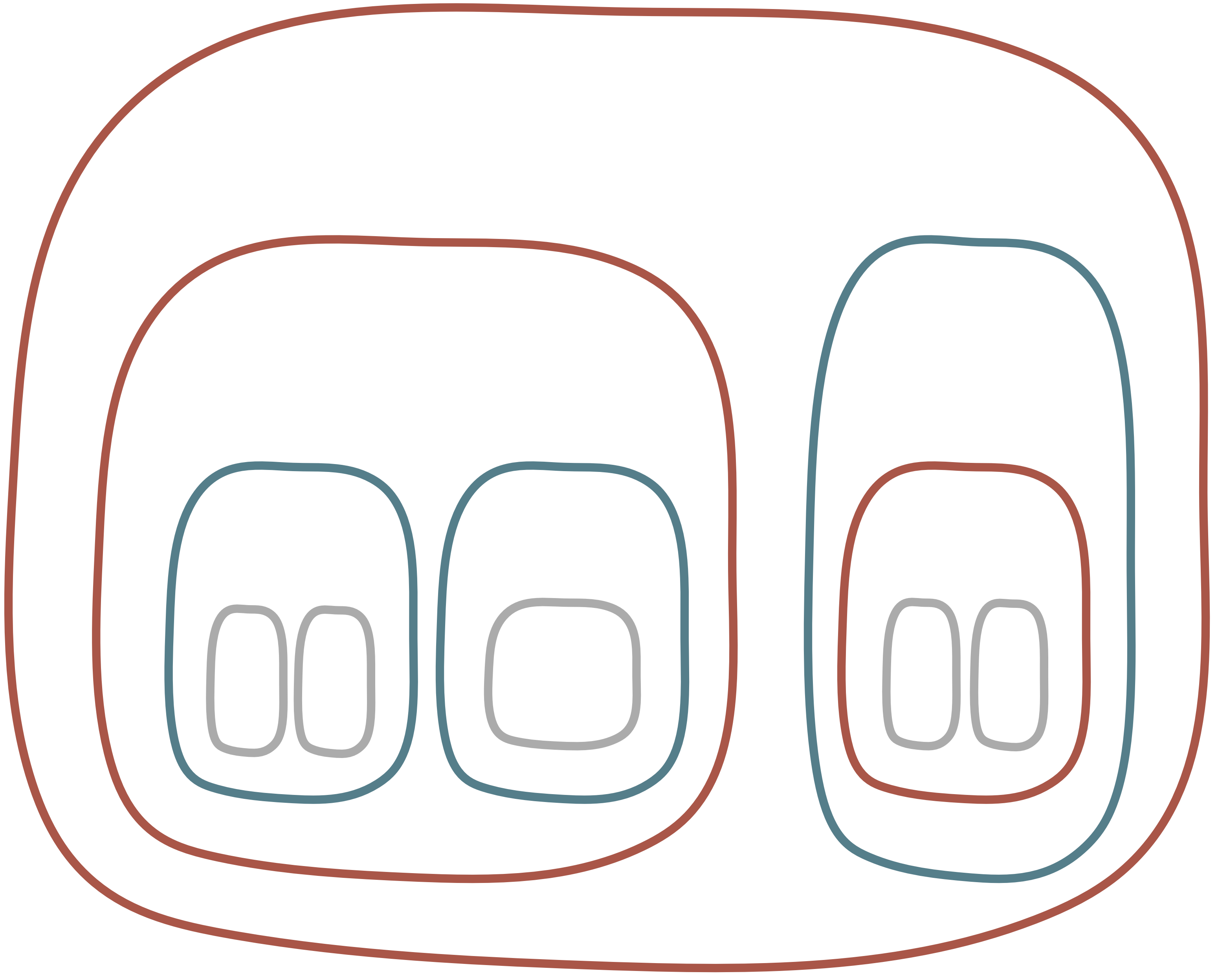


Membrane systems are second class

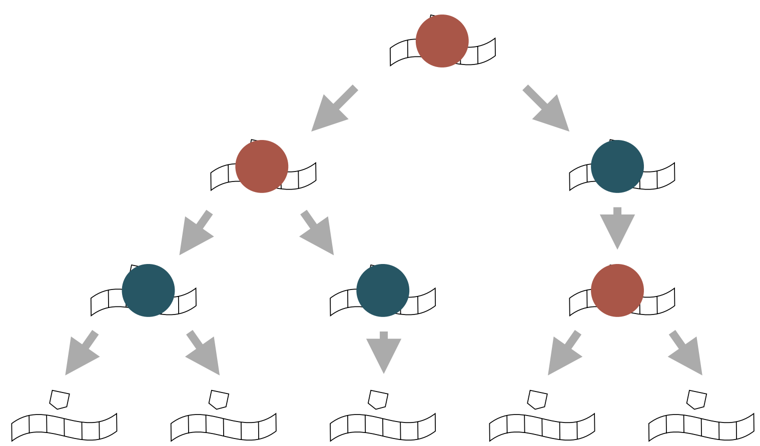


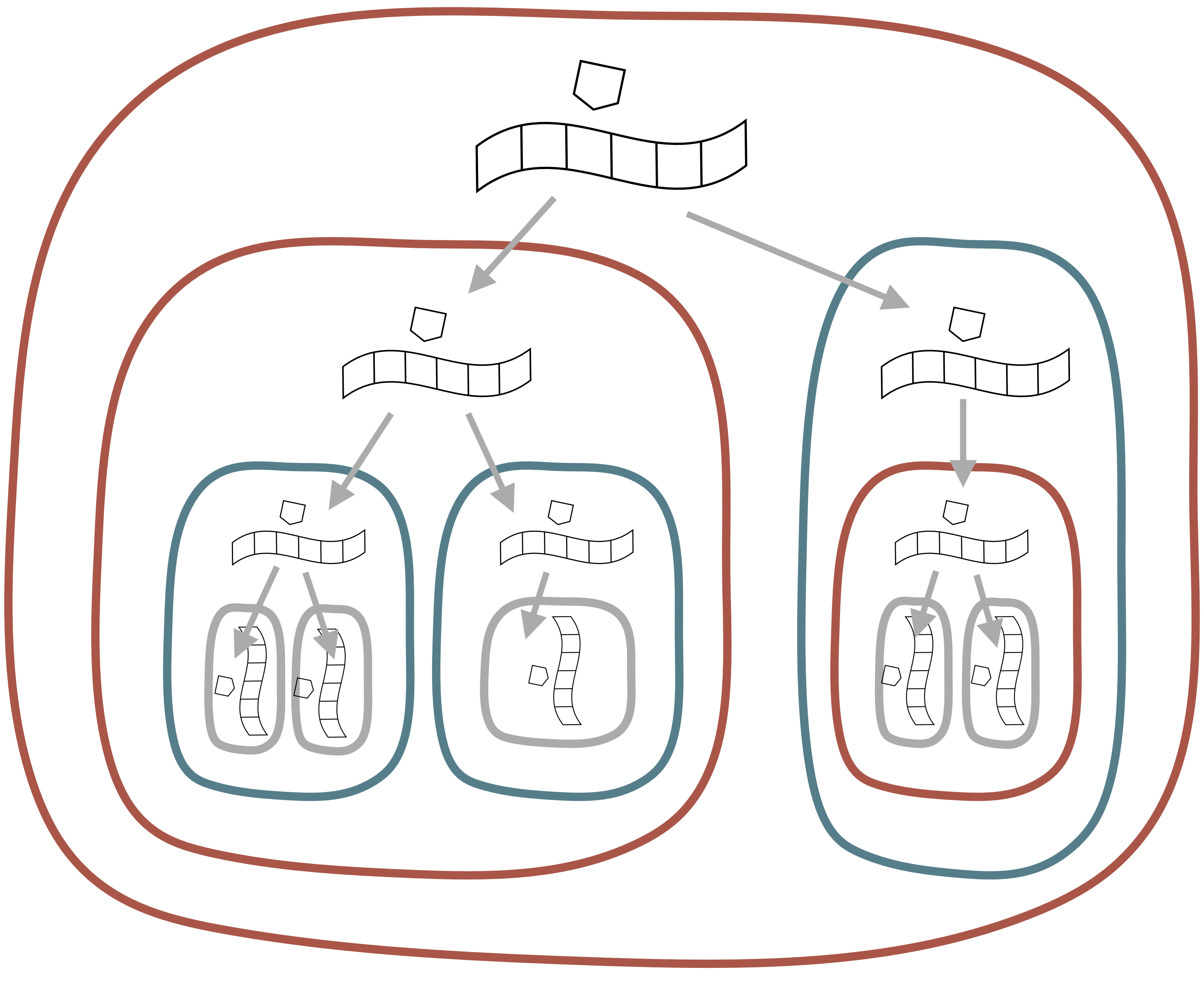
Membrane systems are second class



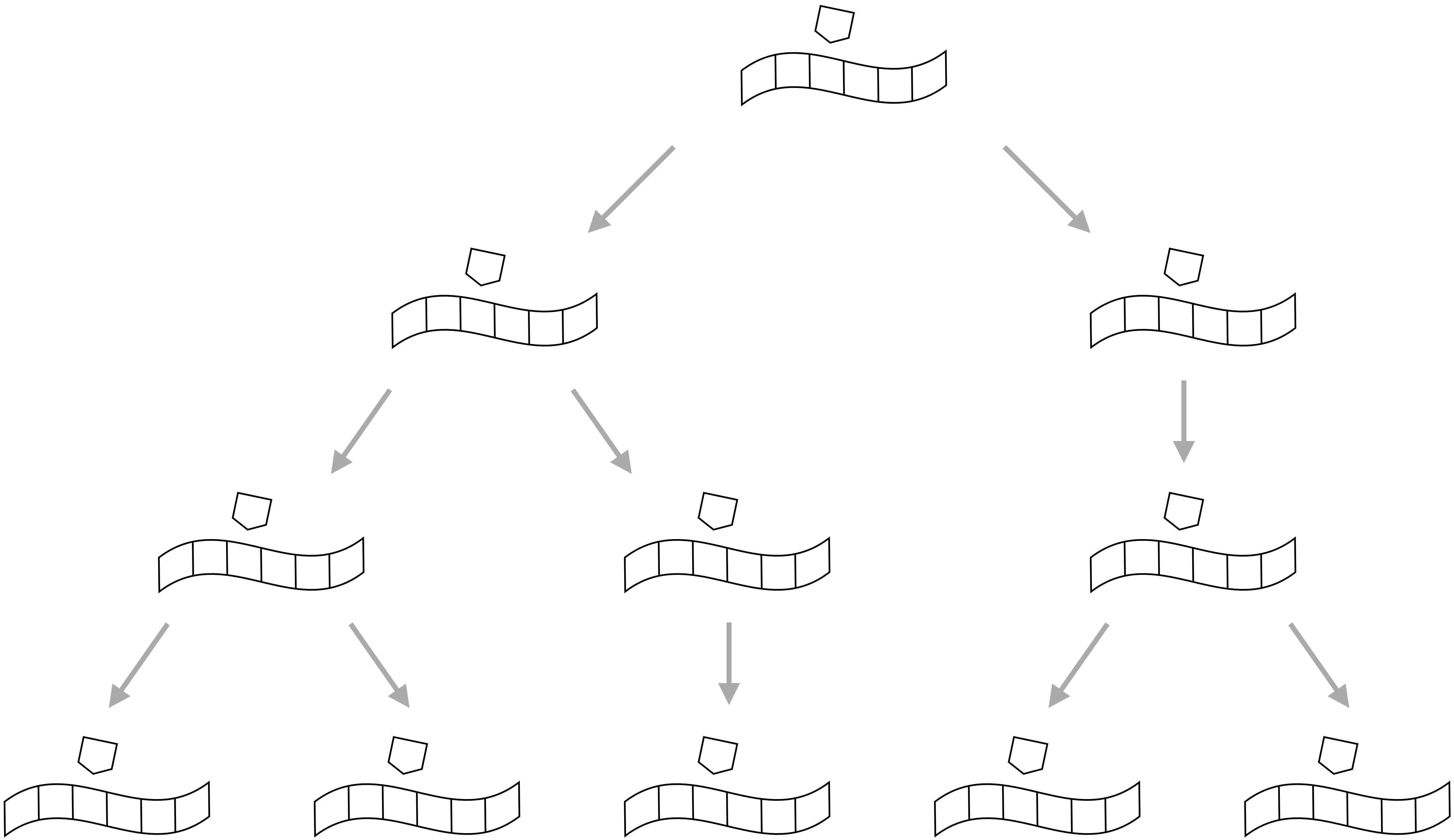


Membrane systems are second class

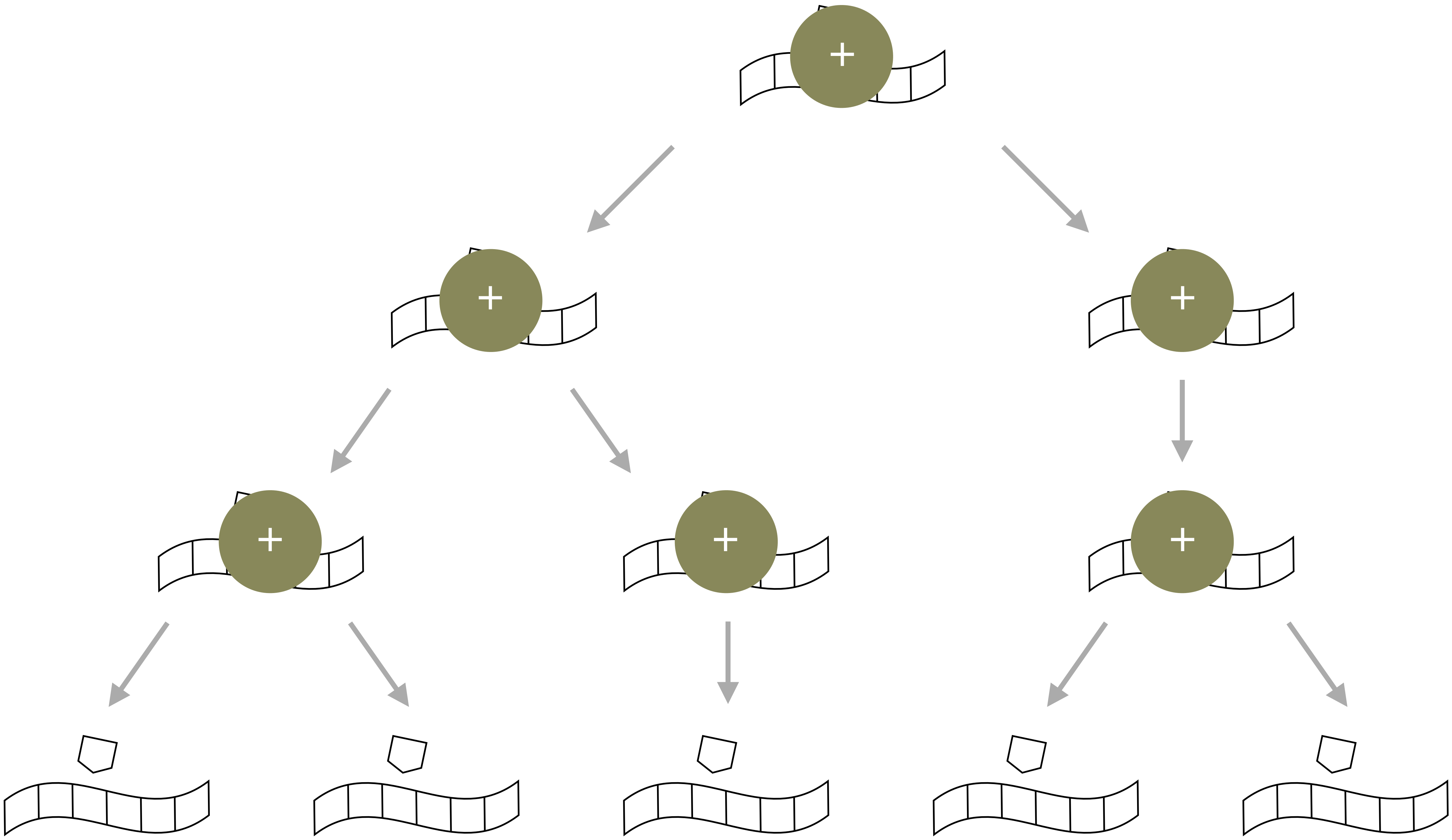




Counting Turing machines: **#P**

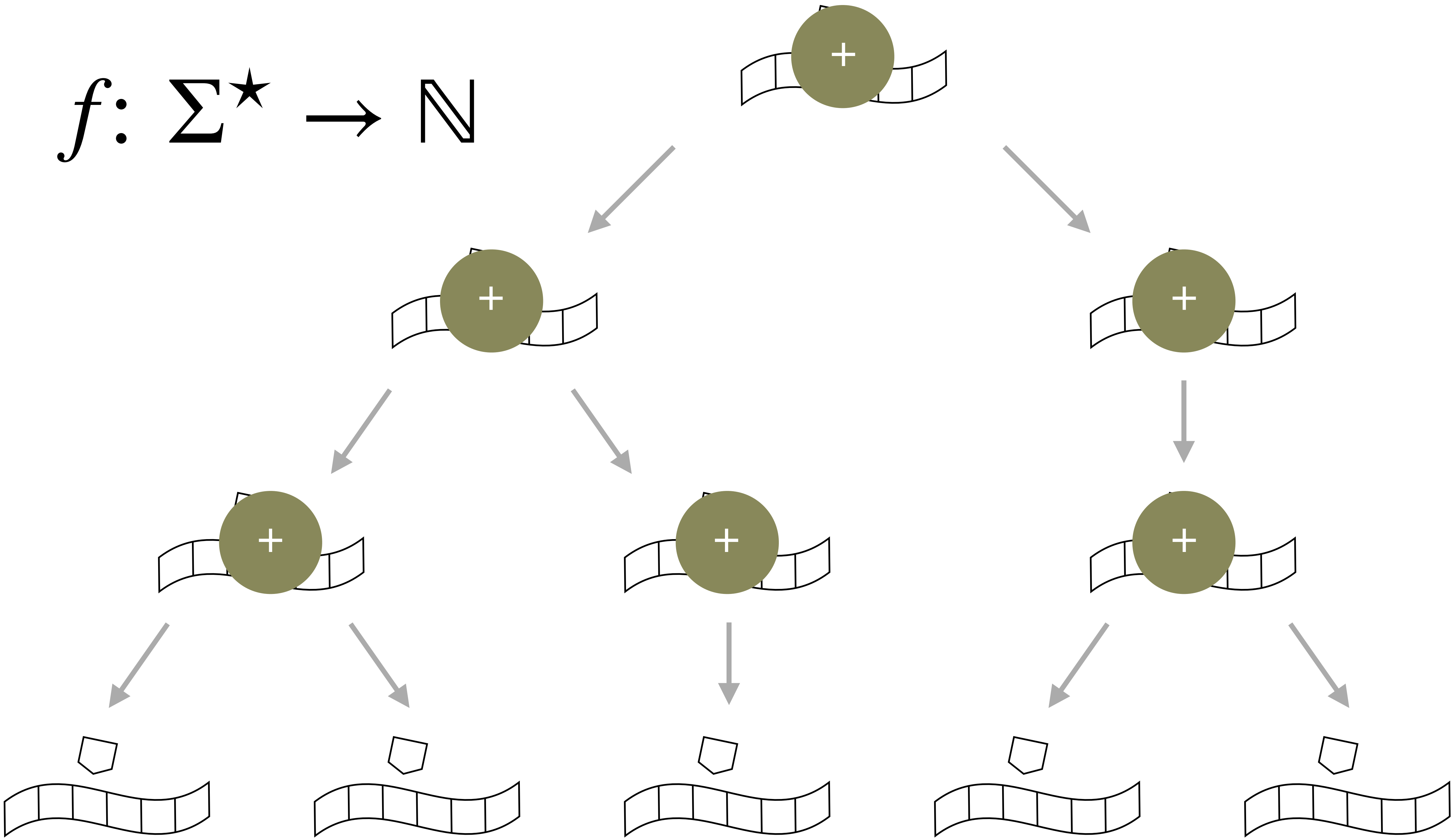


Counting Turing machines: **#P**

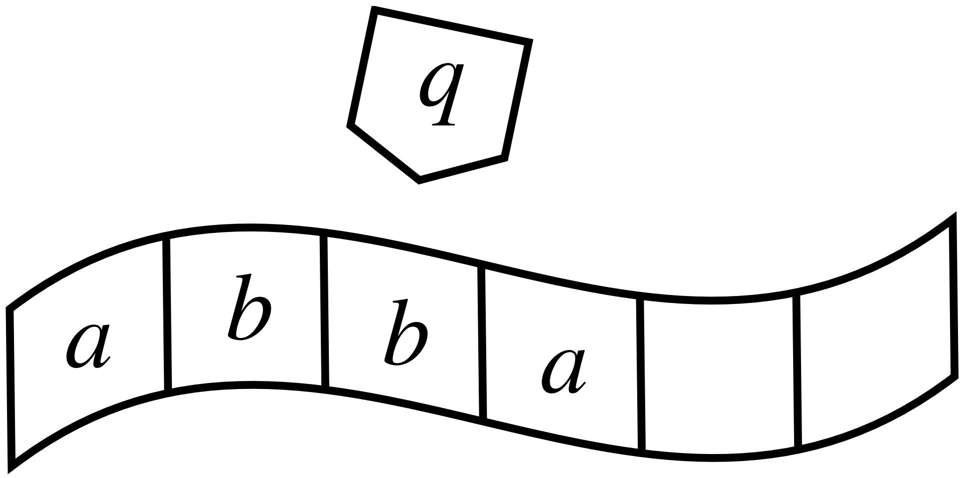


Counting Turing machines: **#P**

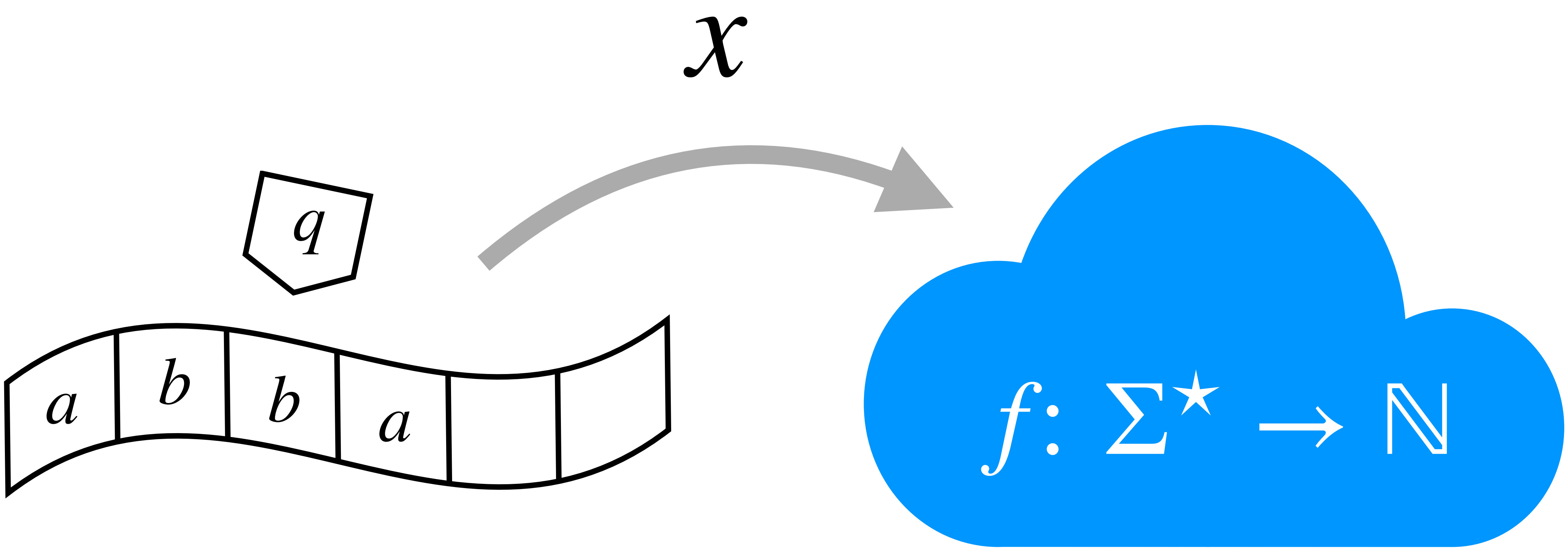
$$f\colon \Sigma^\star \to \mathbb{N}$$



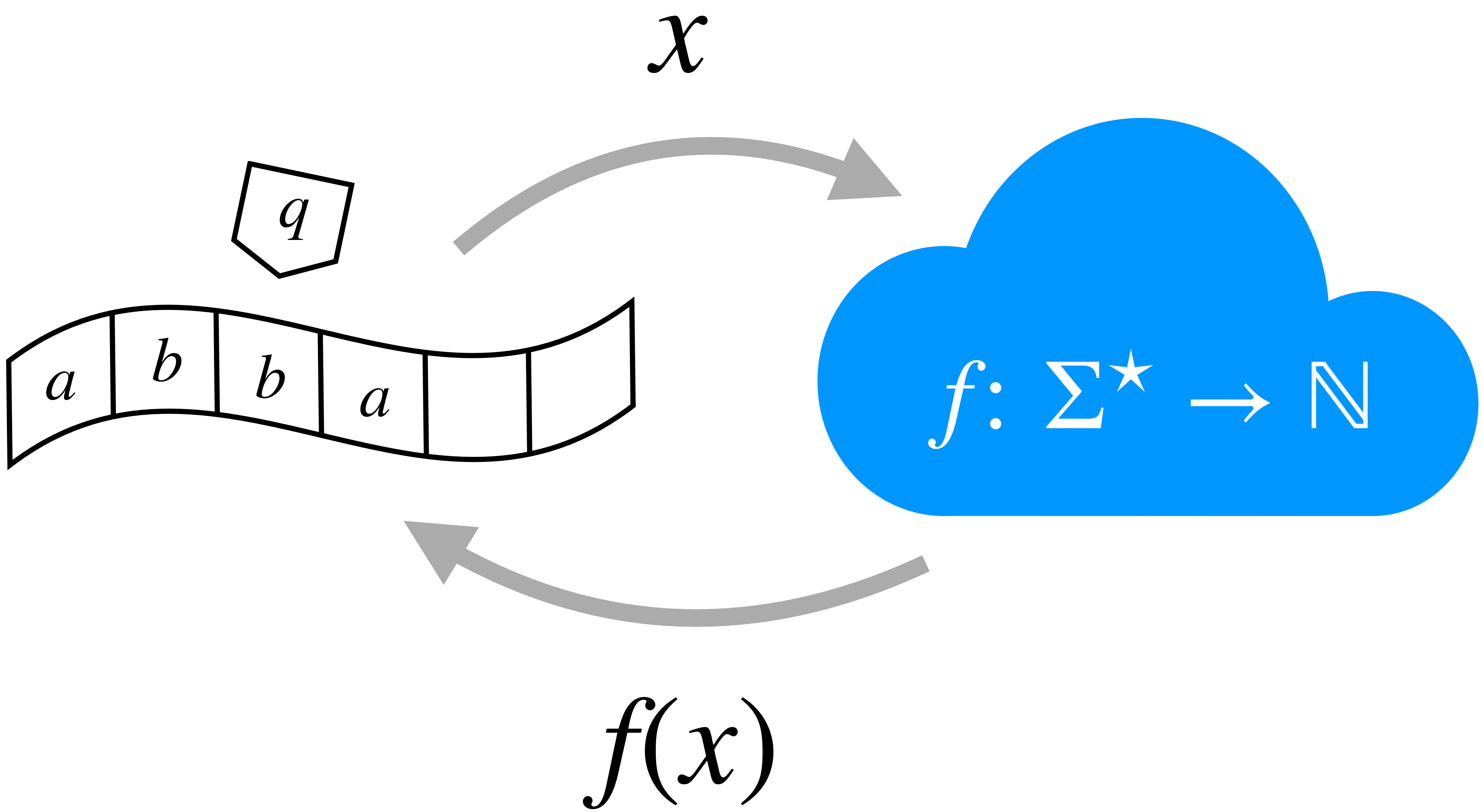
Oracle Turing machines and the class $\mathbf{P}^{\#\mathbf{P}}$



Oracle Turing machines and the class **P#P**



Oracle Turing machines and the class **P**$^{\#\mathbf{P}}$



The class $\mathbf{P}^{\#\mathbf{P}}$ in context

$$\mathbf{P}^{\#\mathbf{P}}$$

The class $\mathbf{P}^{\#\mathbf{P}}$ in context

$$\mathbf{P}^{\#\mathbf{P}}$$

$$\subseteq$$

$$\mathbf{PSPACE}$$

The class $\mathbf{P}^{\mathbf{\#P}}$ in context

$$\overbrace{\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{NP}^{\mathbf{NP}} \subseteq \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \subseteq \cdots}^{\mathbf{PH}}$$

$$\rotatebox{90}{$\subseteq$}$$

$$\mathbf{P}^{\mathbf{\#P}}$$

$$\rotatebox{90}{$\subseteq$}$$

$$\mathbf{PSPACE}$$

The class $\mathbf{P}^{\#\mathbf{P}}$ in context

$$\overbrace{\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{NP}^{\mathbf{NP}} \subseteq \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \subseteq \cdots}^{\mathbf{PH}}$$

$$\rotatebox{90}{$\subseteq$}$$

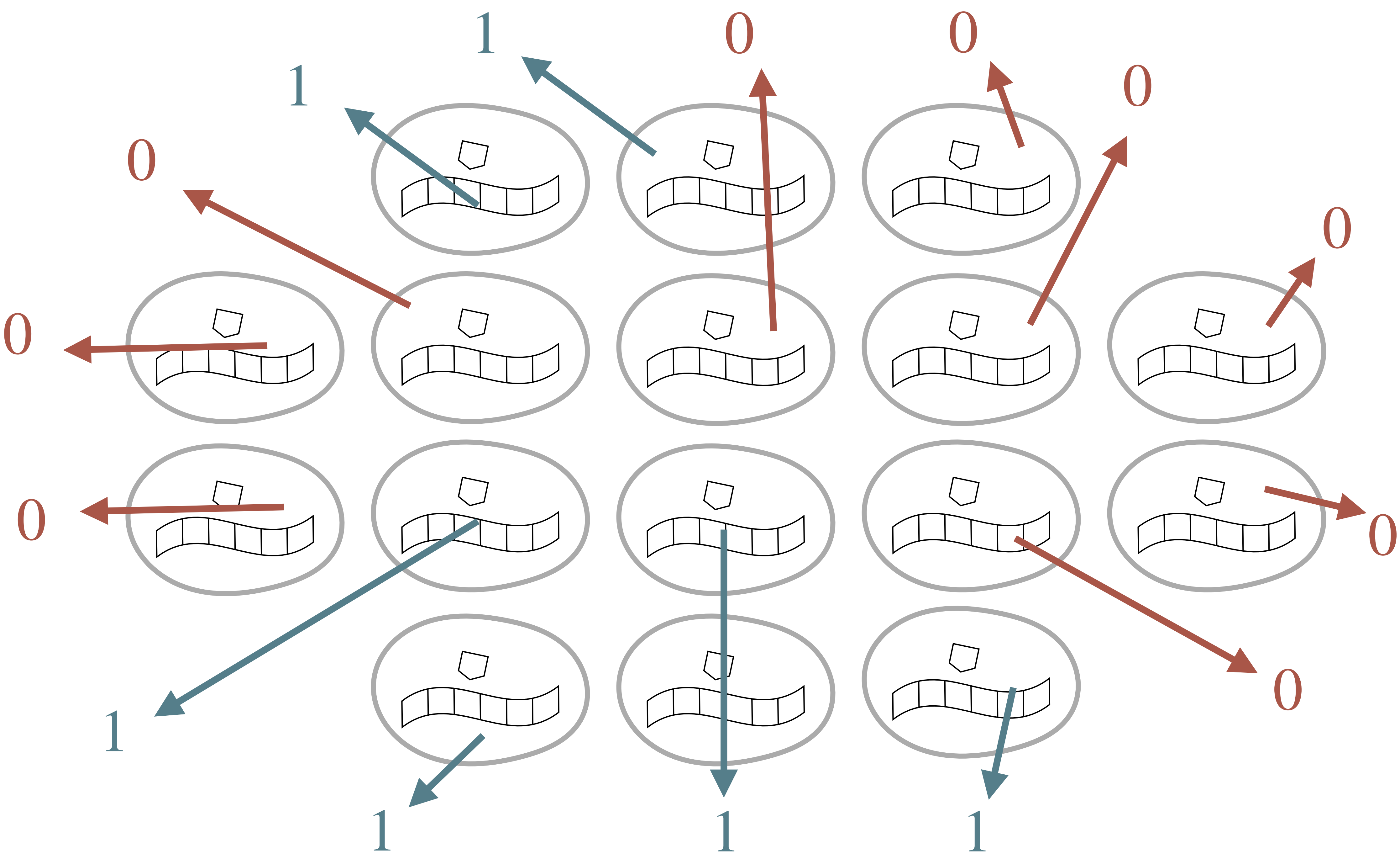
Toda's theorem

$$\mathbf{P}^{\#\mathbf{P}}$$

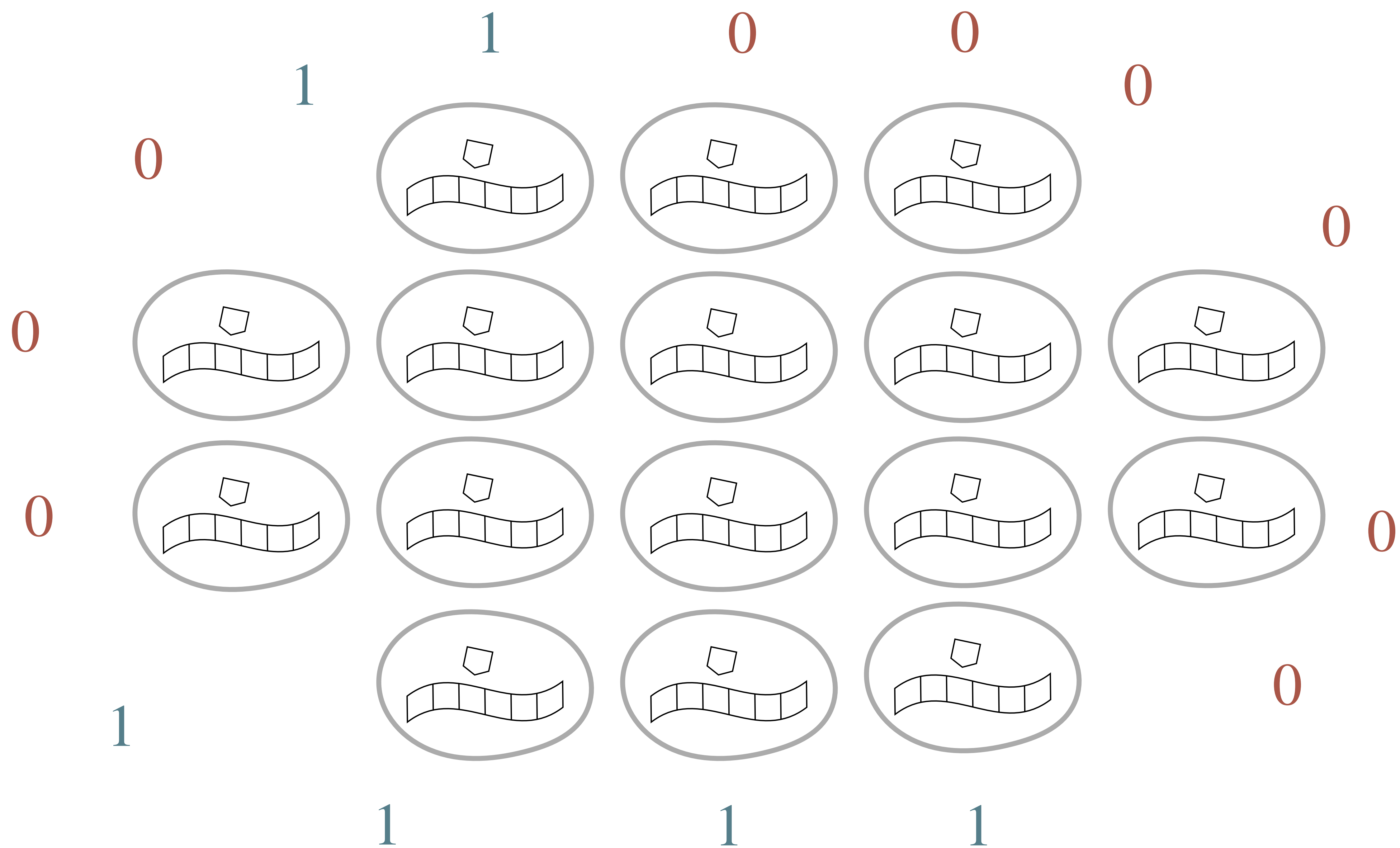
$$\rotatebox{90}{$\subseteq$}$$

$$\mathbf{PSPACE}$$

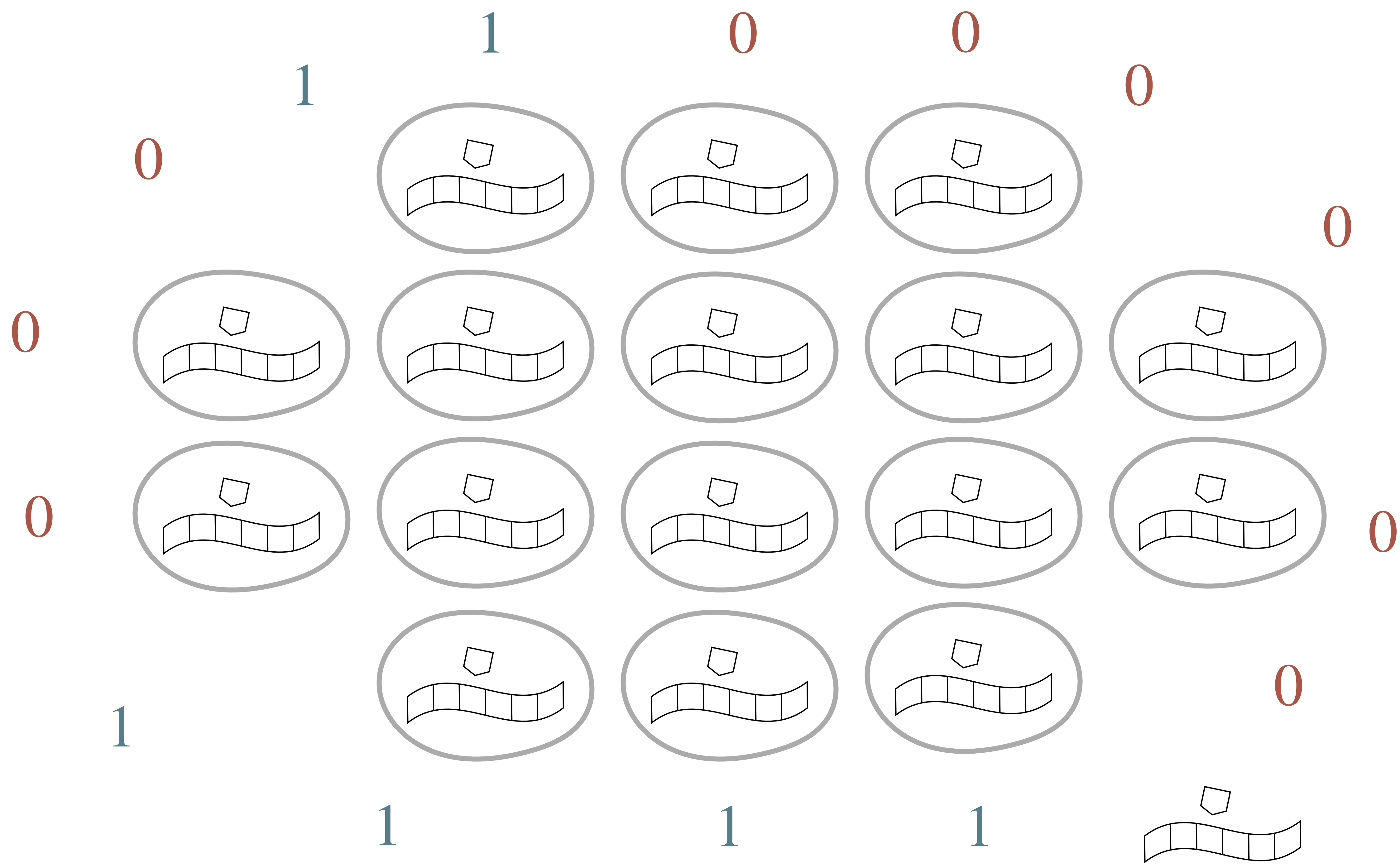
Shallow membrane systems solve **P$^{\#P}$**



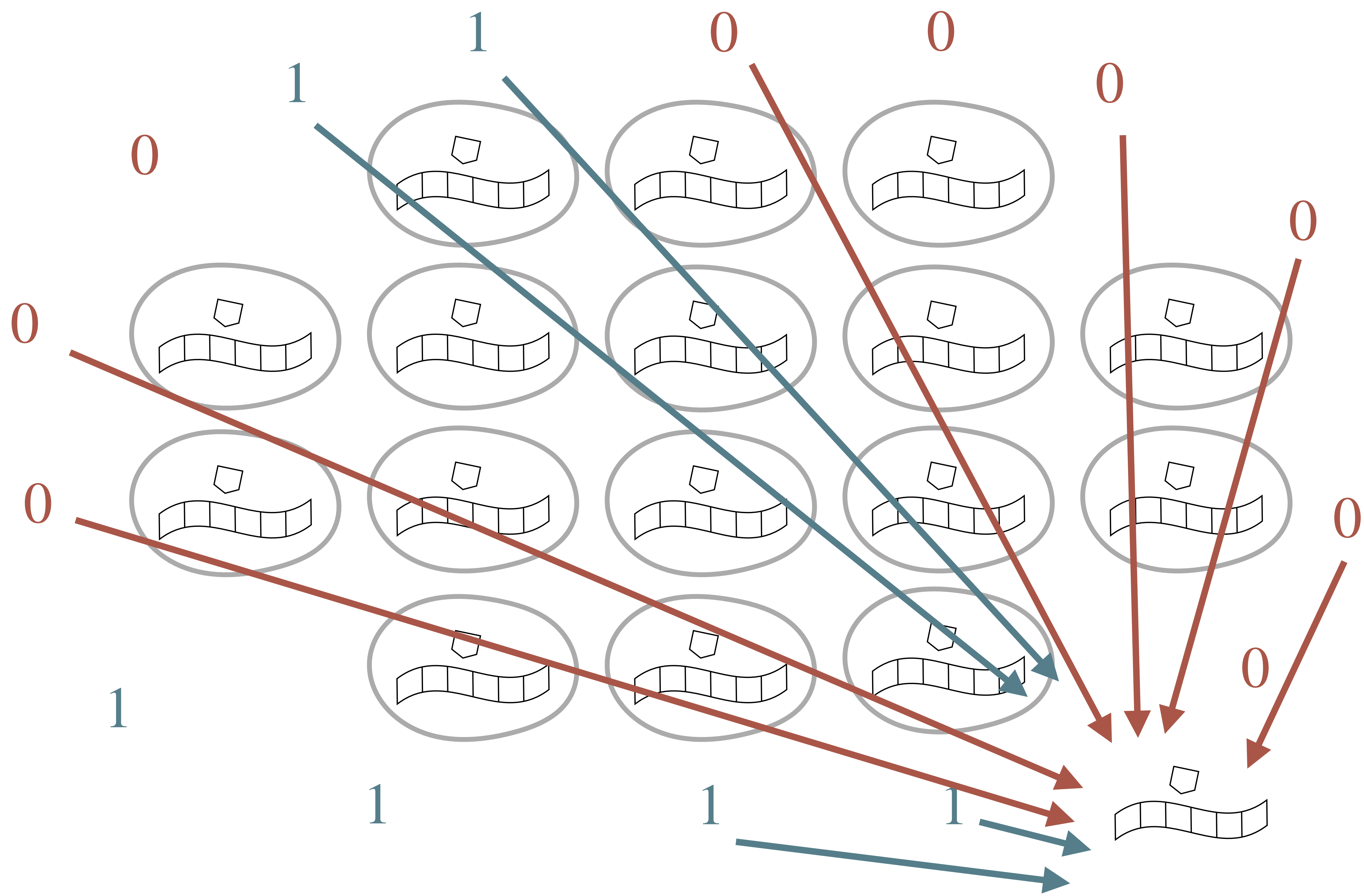
Shallow membrane systems solve $\mathbf{P}^{\mathbf{\#P}}$



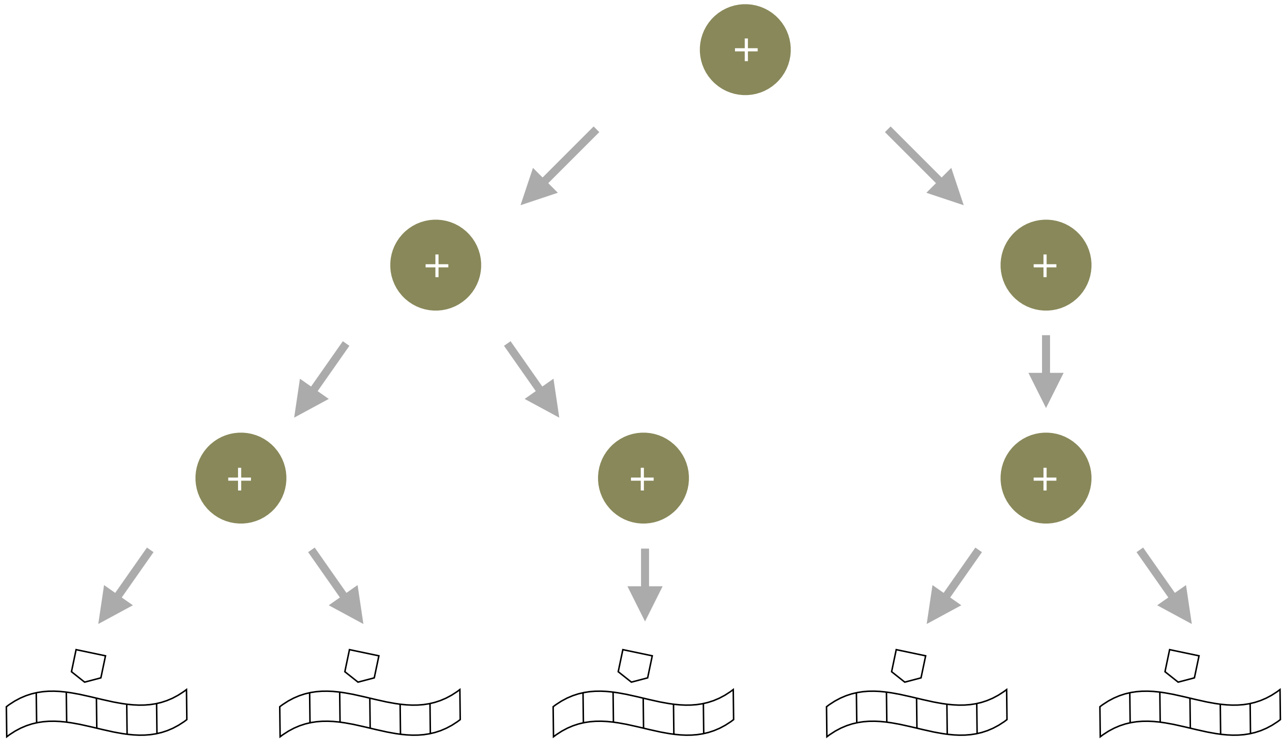
Shallow membrane systems solve $\mathbf{P}^{\#\mathbf{P}}$



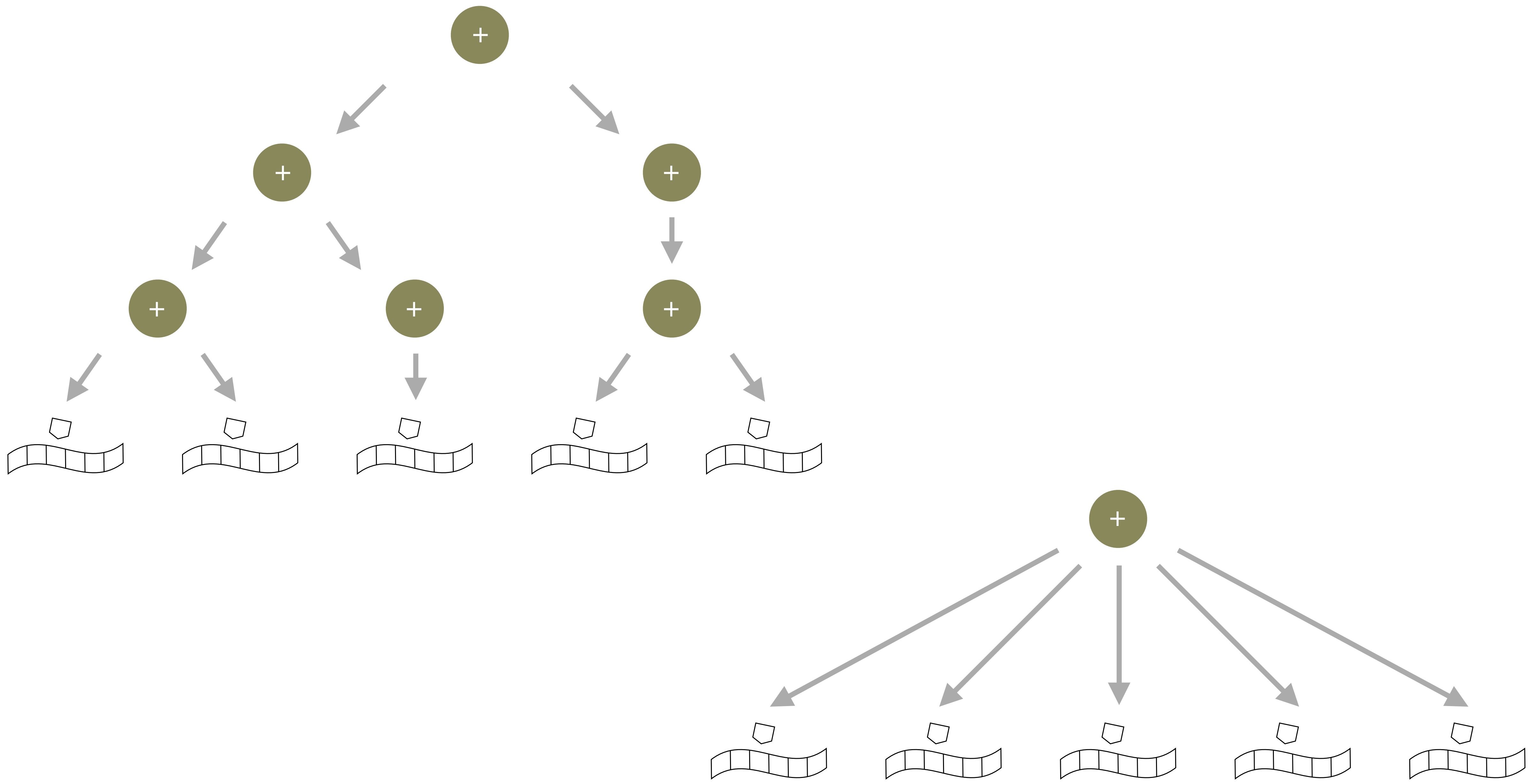
Shallow membrane systems solve $\mathbf{P}^{\#\mathbf{P}}$



Flattening counting computation trees



Flattening counting computation trees



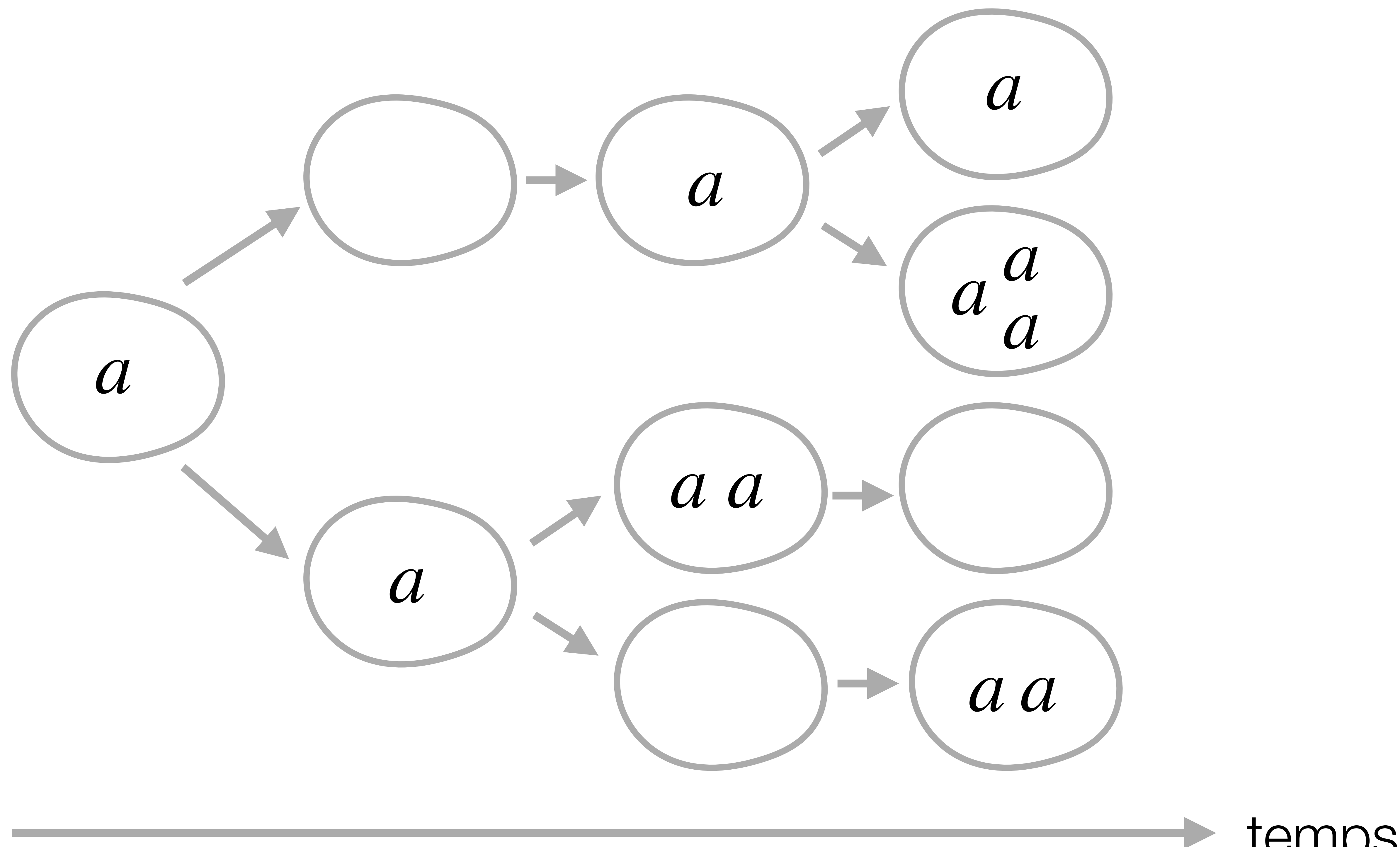
The results up to now

- No membranes (equivalently, no division) = **P**
- Shallow (depth 1) membrane division ⊇ $\mathbf{P}^{\#\mathbf{P}}$
- Deep membrane division = **PSPACE**

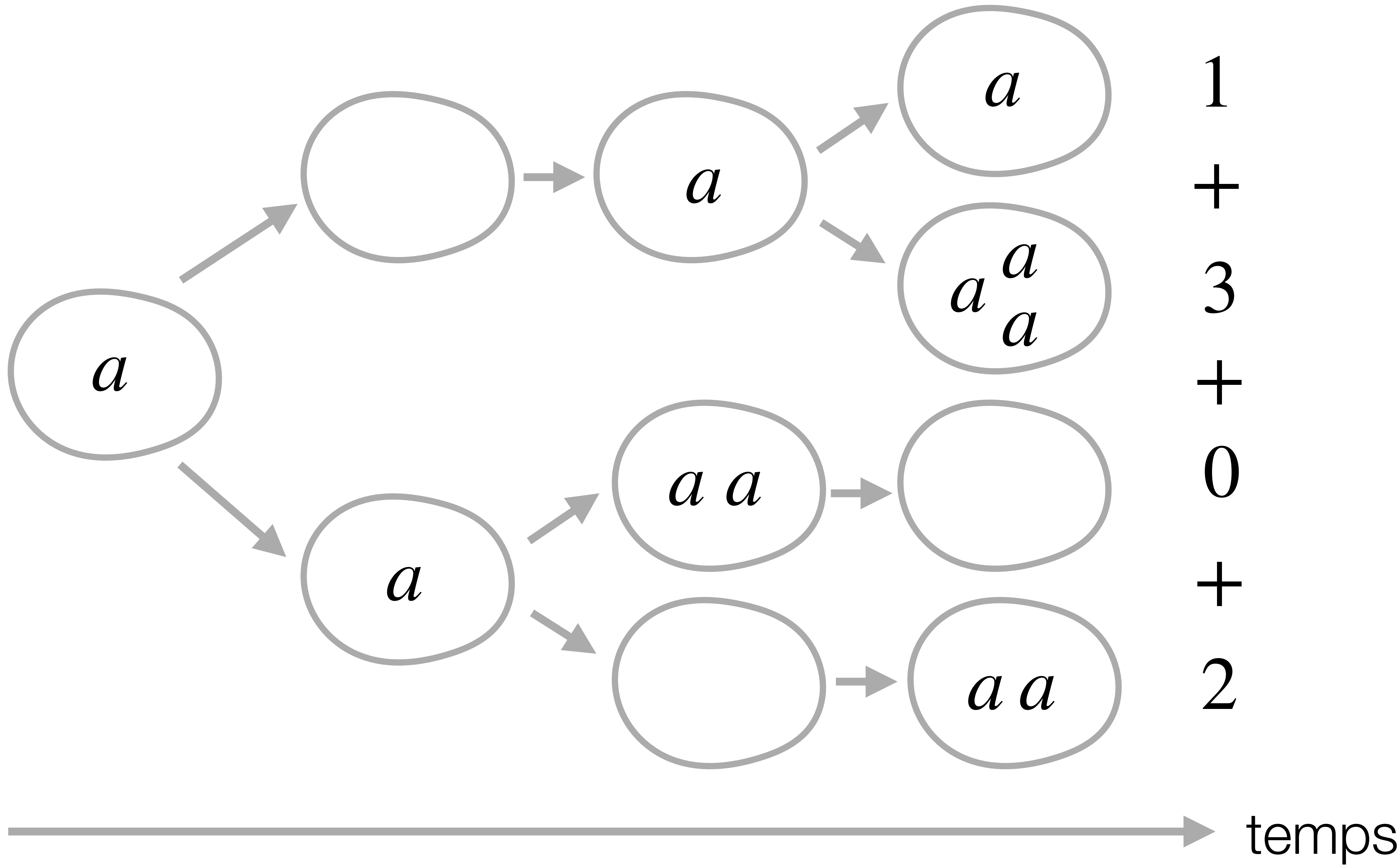
Proving that shallow membrane division = $\mathbf{P}^{\#\mathbf{P}}$

Given the initial configuration
of an elementary membrane,
how many copies of object *a* are sent out
by descendant membranes at time *t*?

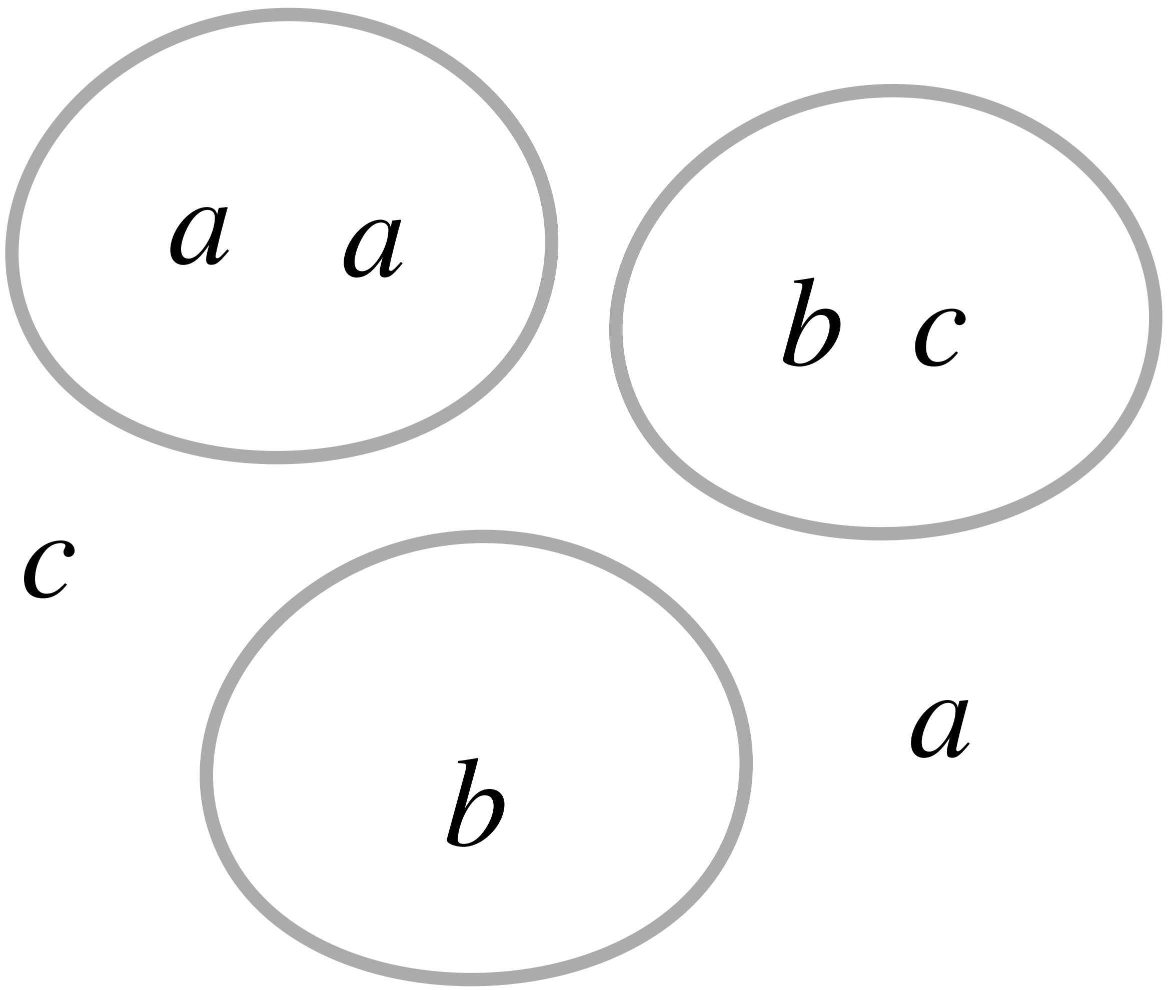
Query in **#P** in the monodirectional case



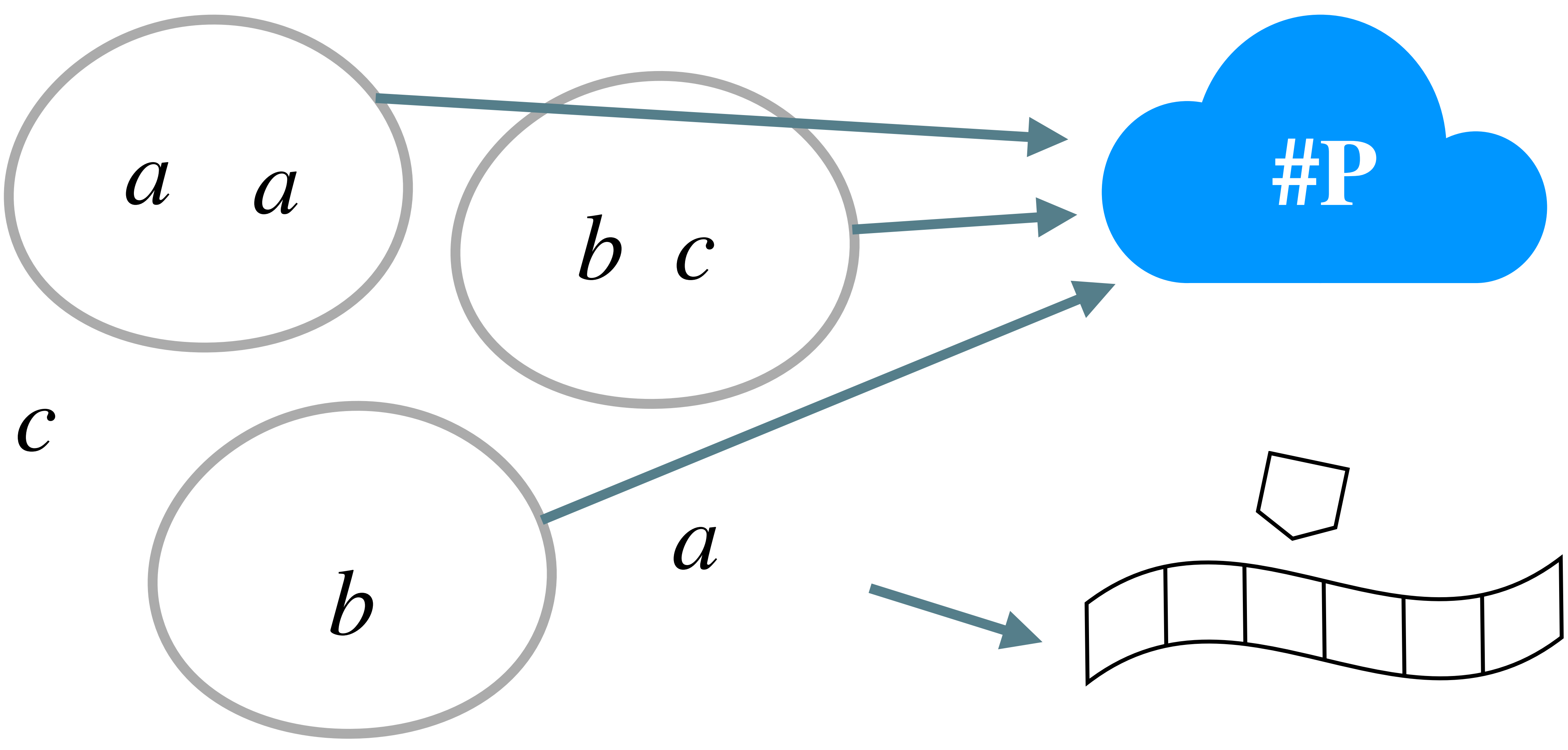
Query in **#P** in the monodirectional case



“Easy” upper bound in the monodirectional case



“Easy” upper bound in the monodirectional case



Simulation algorithm (monodirectional case)

- For each time step *t* until the simulated system halts:
  - Simulate one step of the external environment, updating the stored configuration
  - **Ask the oracle** how many instances of each symbol are sent out at time *t* by each elementary membrane in the initial configuration (not stored)
  - Add the answer to each query to the external environment
- Accept or reject depending on the result of the simulation

“Send-in” communication breaks the algorithm

Given the initial configuration of an elementary membrane and a table describing what communication rules (with multiplicity!) were applied to its descendants until time $t - 1$, how many copies of object a are sent out at time t?

Monodirectional characterisation of $\mathbf{P}^{\mathbf{NP}}$

- Consider membrane systems where deep membrane division is allowed
- But limit inter-membrane communication to monodirectional (outwards)
- Then we obtain a precise characterisation of $\mathbf{P}^{\mathbf{NP}}$ in polynomial time

Finally some unusual complexity classes!

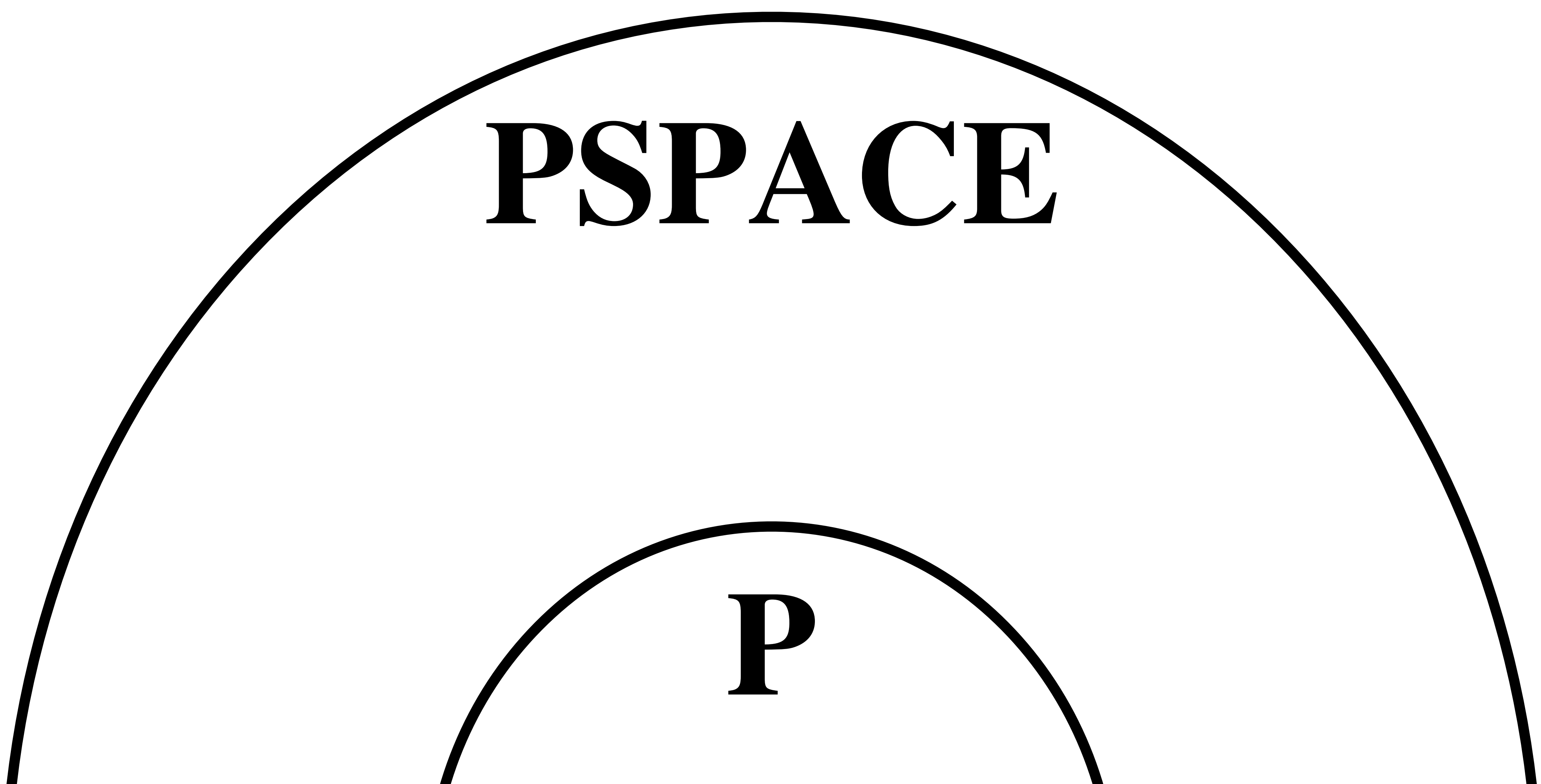
$$\mathbf{P}^{\mathbf{NP}}$$

$$\mathbf{P}^{\#\mathbf{P}}$$

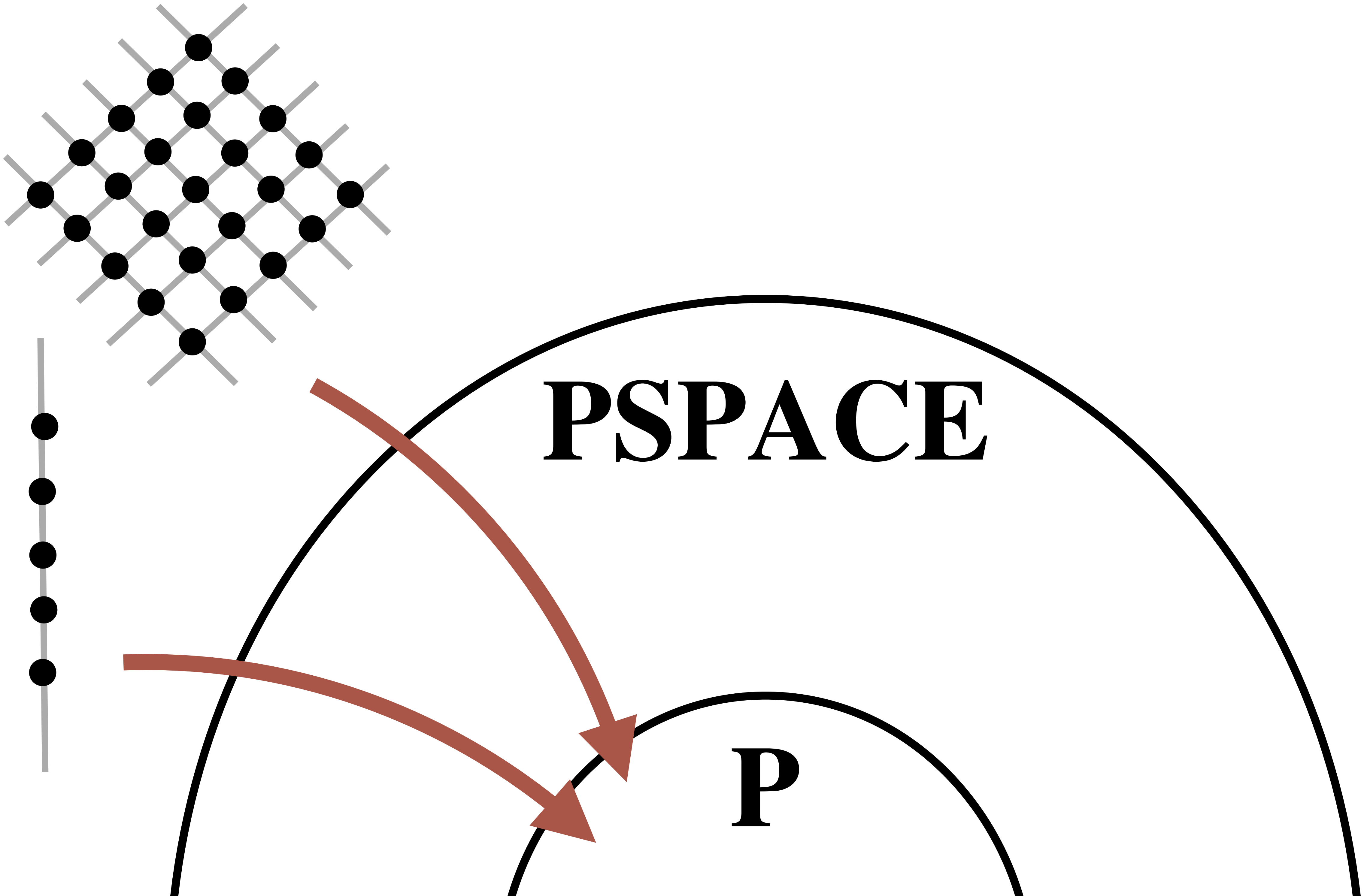
Hierarchy of membrane systems wrt division depth

- No membranes (equivalently, no division) = **P**
- Shallow (depth 1) membrane division = $\mathbf{P}^{\#\mathbf{P}}$
- Deep membrane division = **PSPACE**

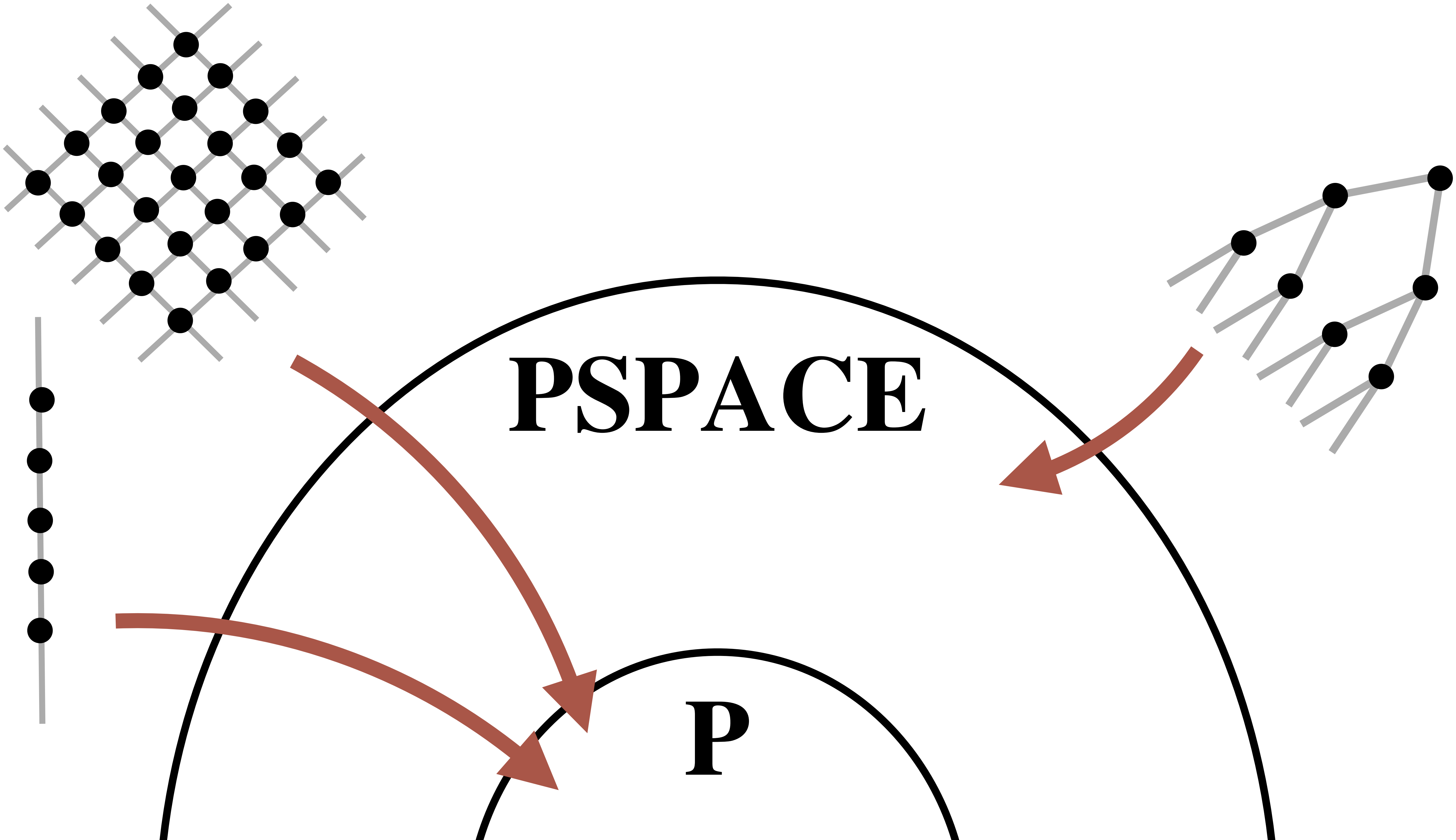
Communication topology and complexity



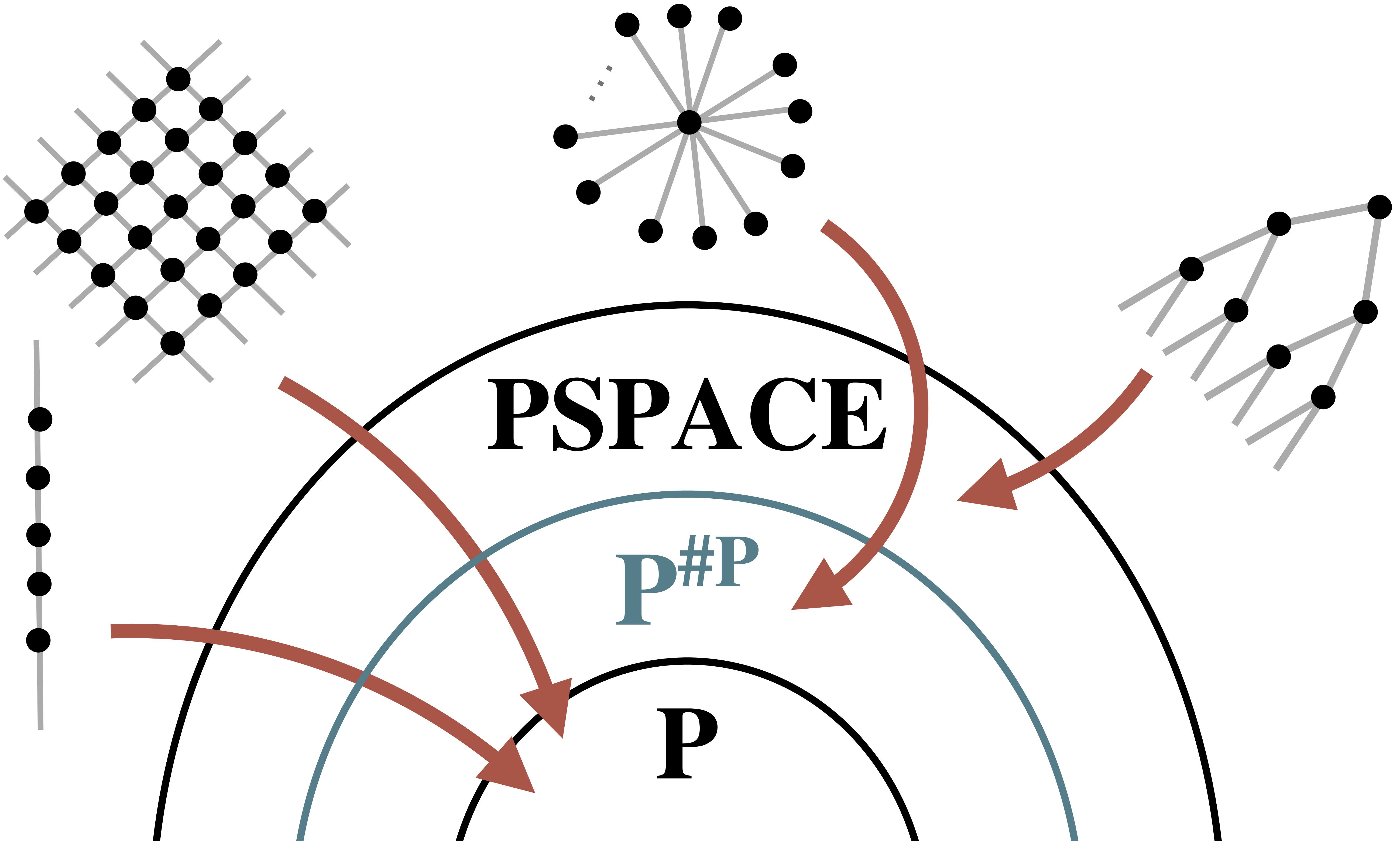
Communication topology and complexity



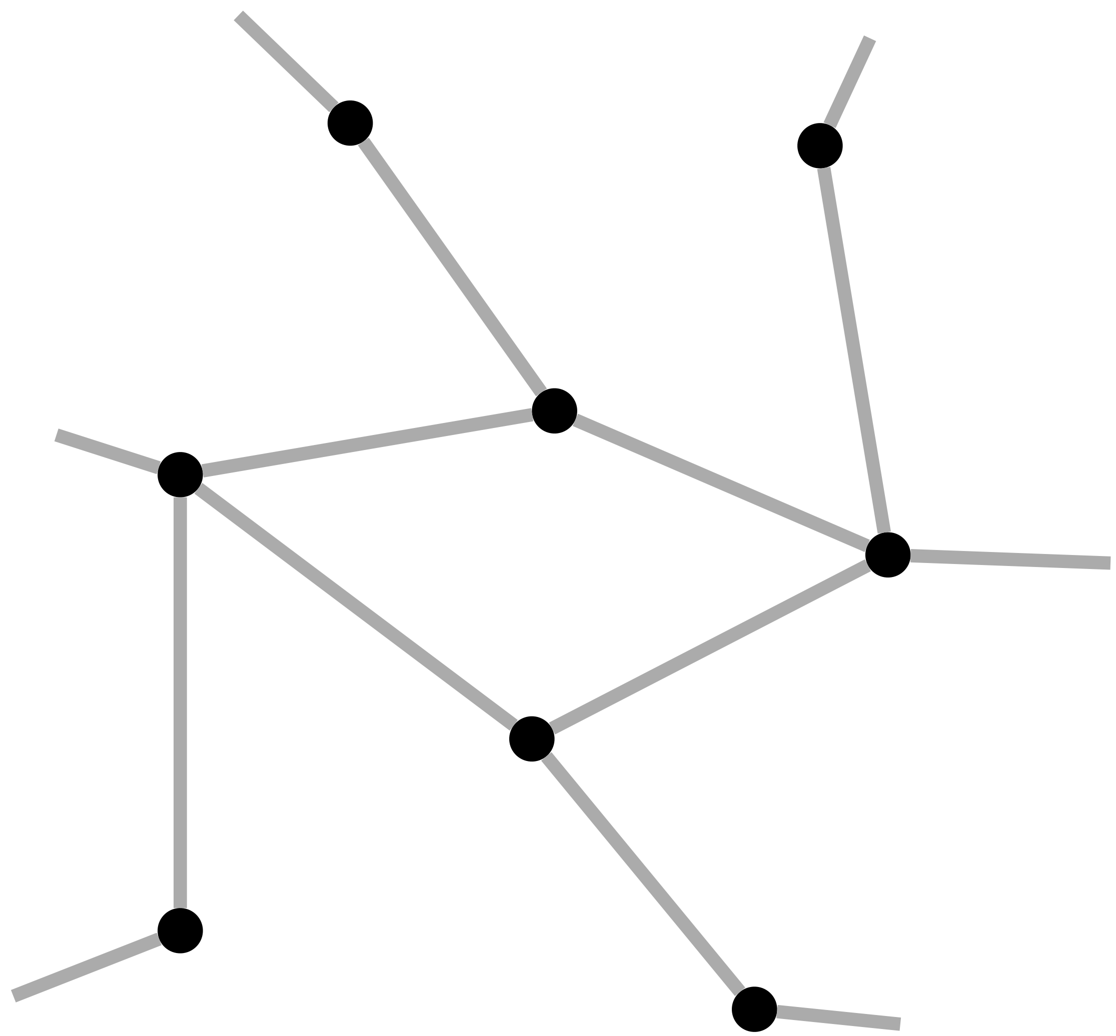
Communication topology and complexity



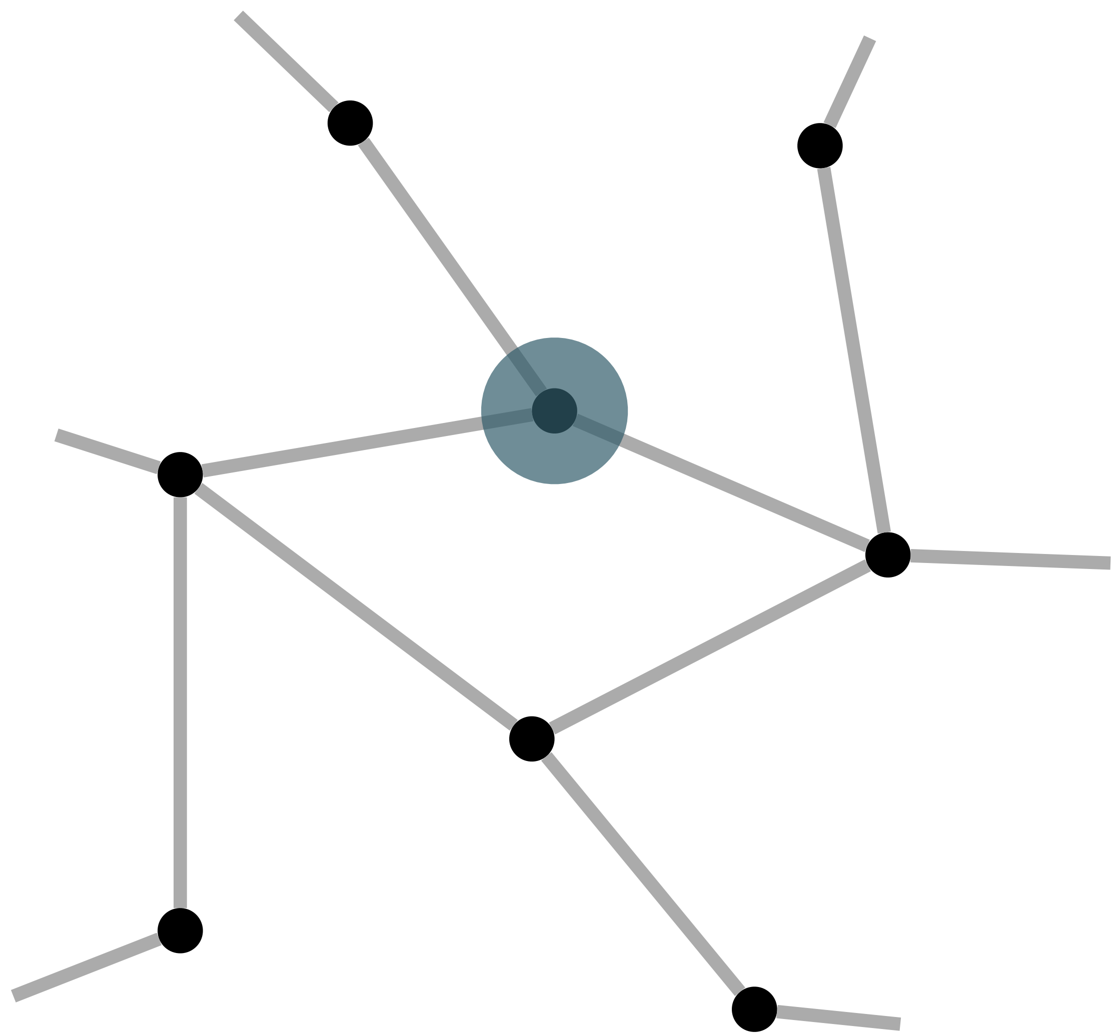
Communication topology and complexity



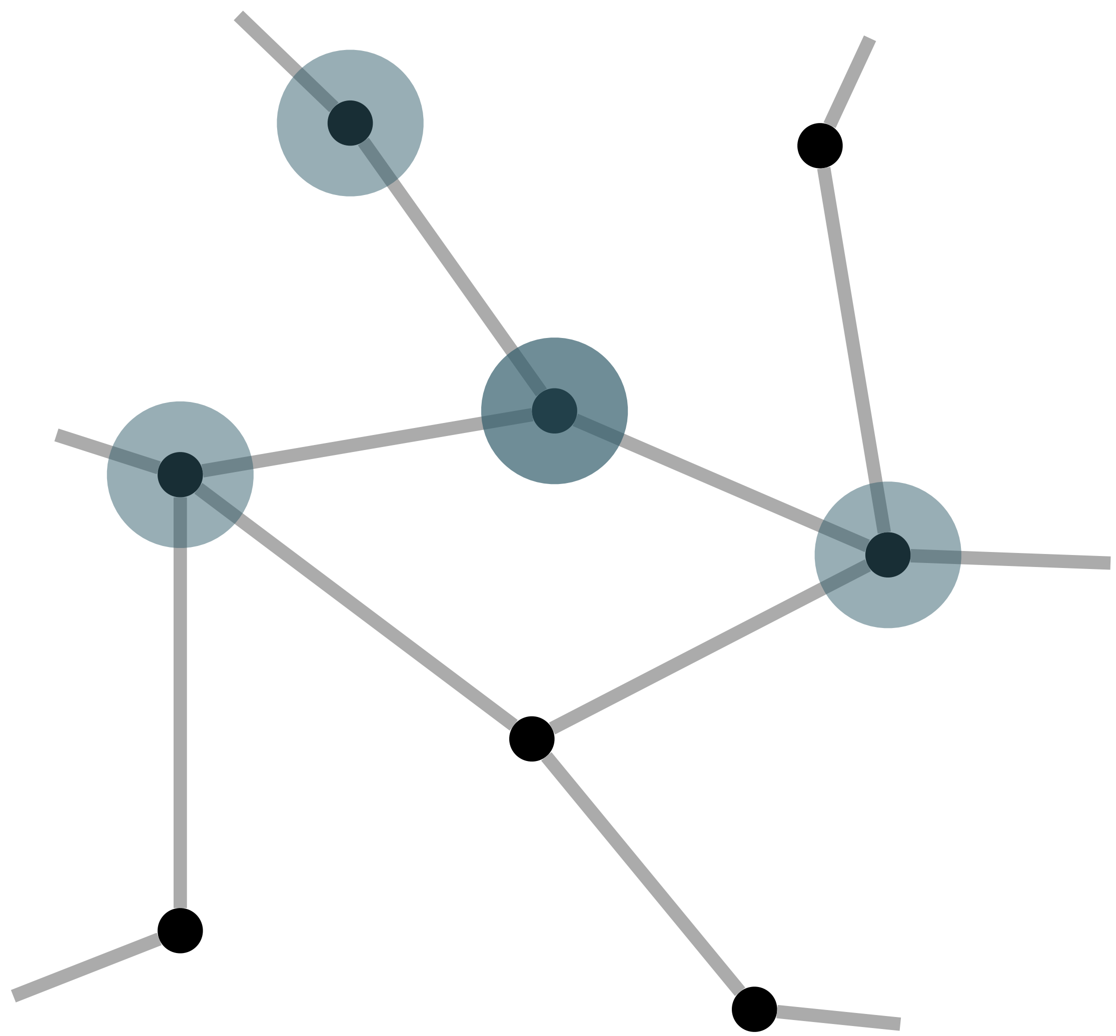
Automata networks over arbitrary infinite graphs as a reference model of computation



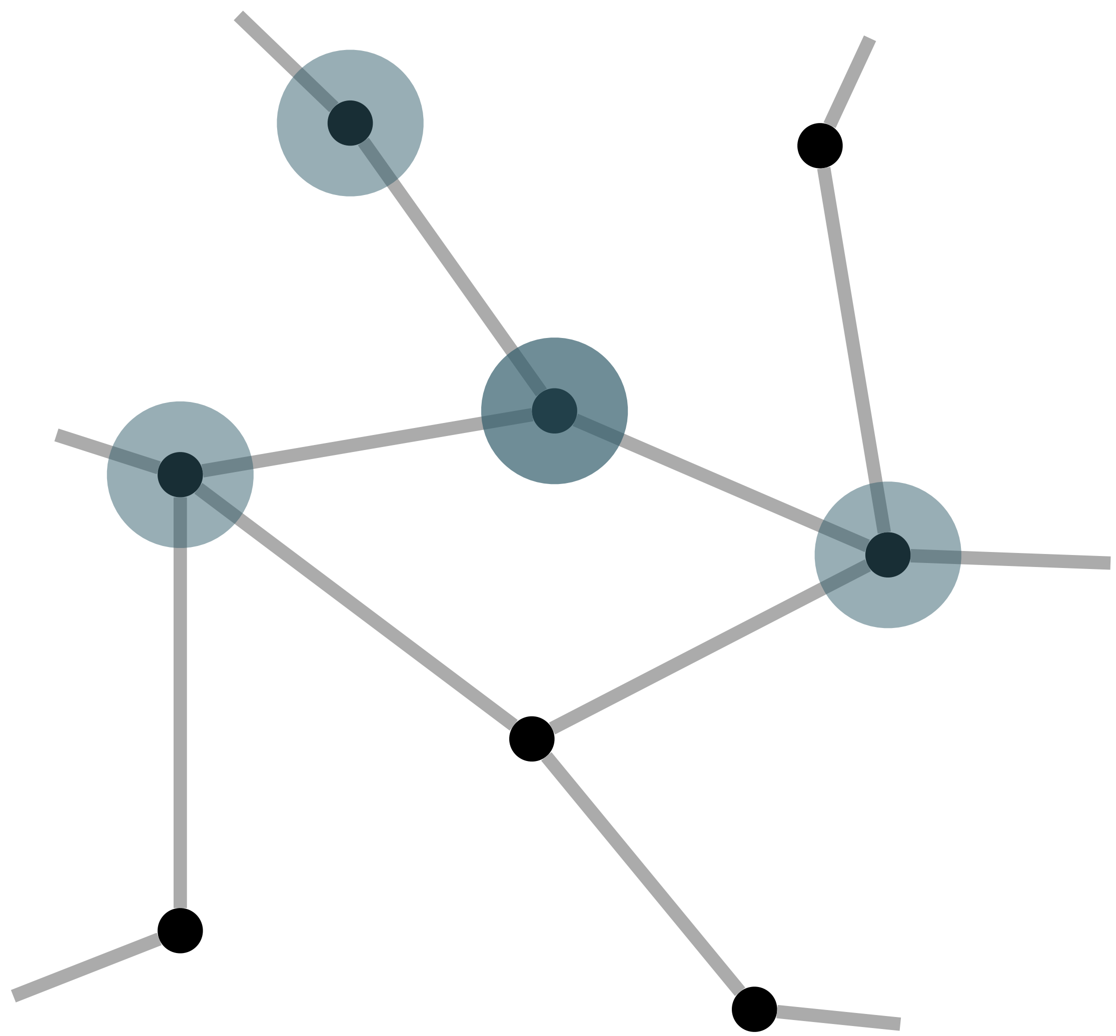
Automata networks over arbitrary infinite graphs as a reference model of computation



Automata networks over arbitrary infinite graphs as a reference model of computation



Automata networks over arbitrary infinite graphs as a reference model of computation



$$f\colon \mathrm{Multisets}(Q) \to Q$$

Generalised complexity classes over a graph G

$\mathbf{PSPACE}(G)$

$\mathbf{P}(G)$

$\mathbf{EXPTIME}(G)$

$\mathbf{LOGTIME}(G)$

$\mathbf{NP}(G)$

Expected results

$$\mathbf{P}(\text{Euclidean grid}) = \mathbf{P}$$

$$\mathbf{P}(\text{infinite binary tree}) = \mathbf{PSPACE}$$

$$\mathbf{P}(\text{variant of infinite star}) = \mathbf{P}^{\#\mathbf{P}}$$

Open problems

- Find explanations to the membrane computing results
- Find graphs characterising the standard complexity classes
- Find complexity classes corresponding to “natural” families of graphs (i.e., with interesting graph-theoretic properties)
- Find how the geometry of the space influence the efficiency of the algorithms

Thanks for your attention!
Merci de votre attention !

Any questions?