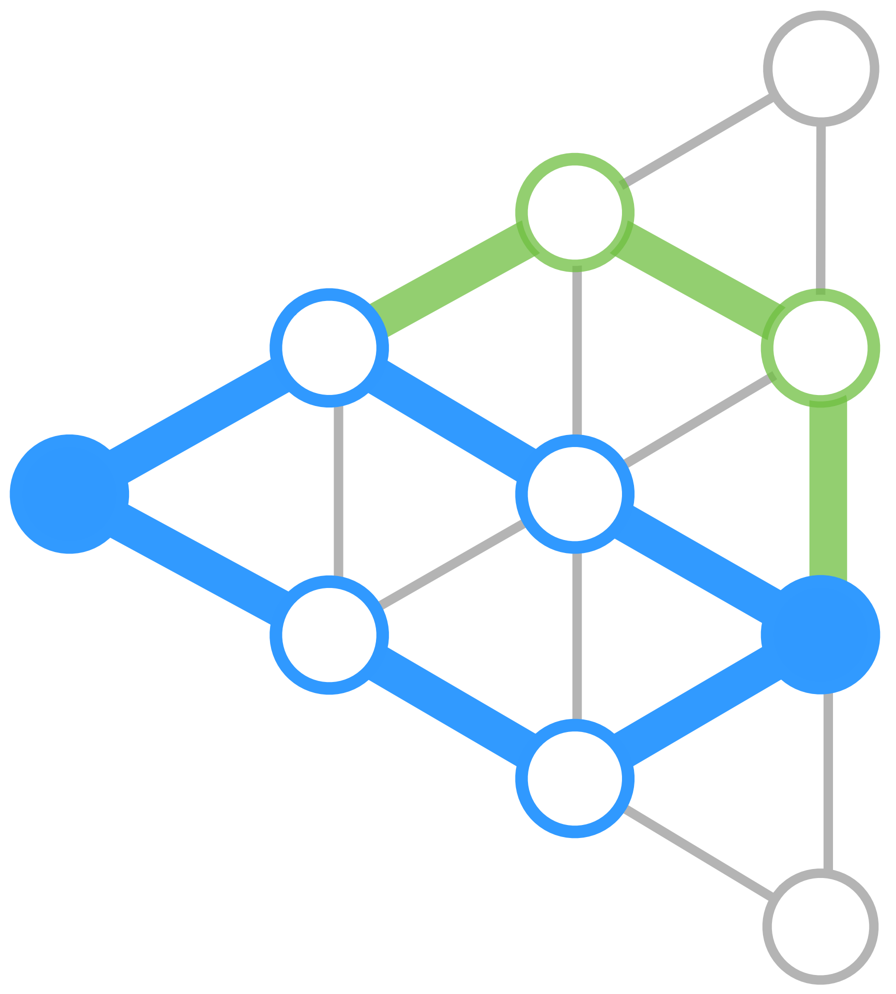


# Approximate agreement on graphs revisited

**ADGA 2023**

**Joel Rybicki**
Humboldt University of Berlin

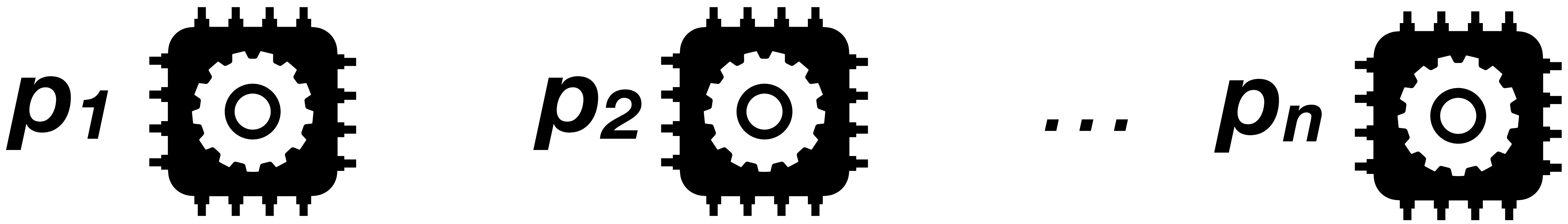
October 9, 2023

**Based on joint work with:**
Dan Alistarh
Faith Ellen
Thomas Nowak

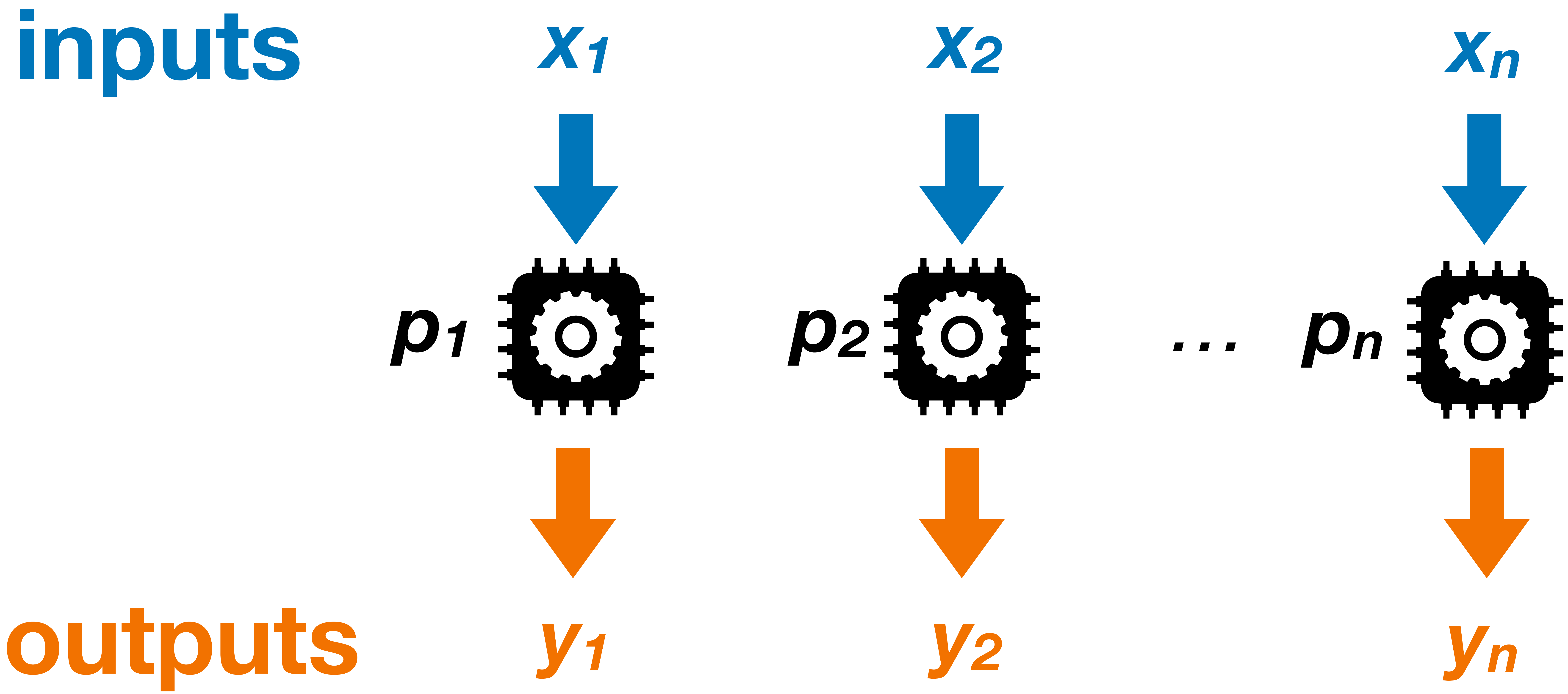
# Outline

1. Approximate agreement on graphs
2. Wait-free shared memory algorithms
3. Impossibility results
4. Open problems

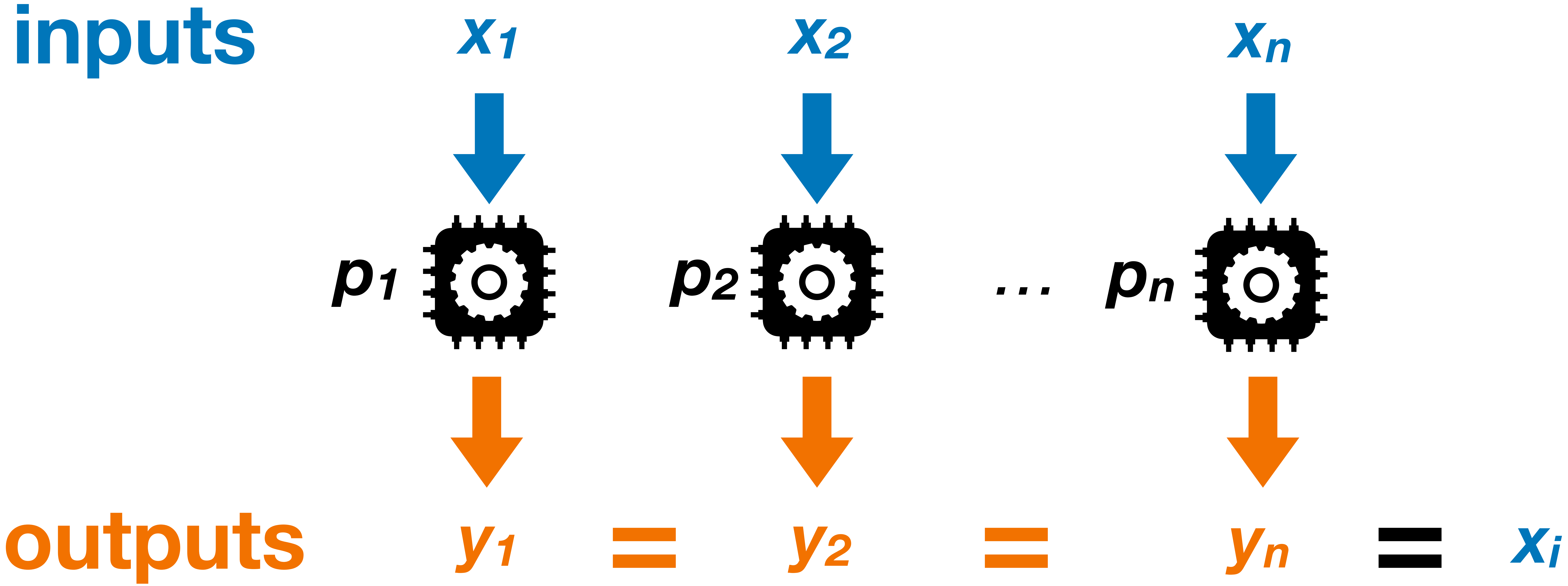
# Distributed agreement tasks



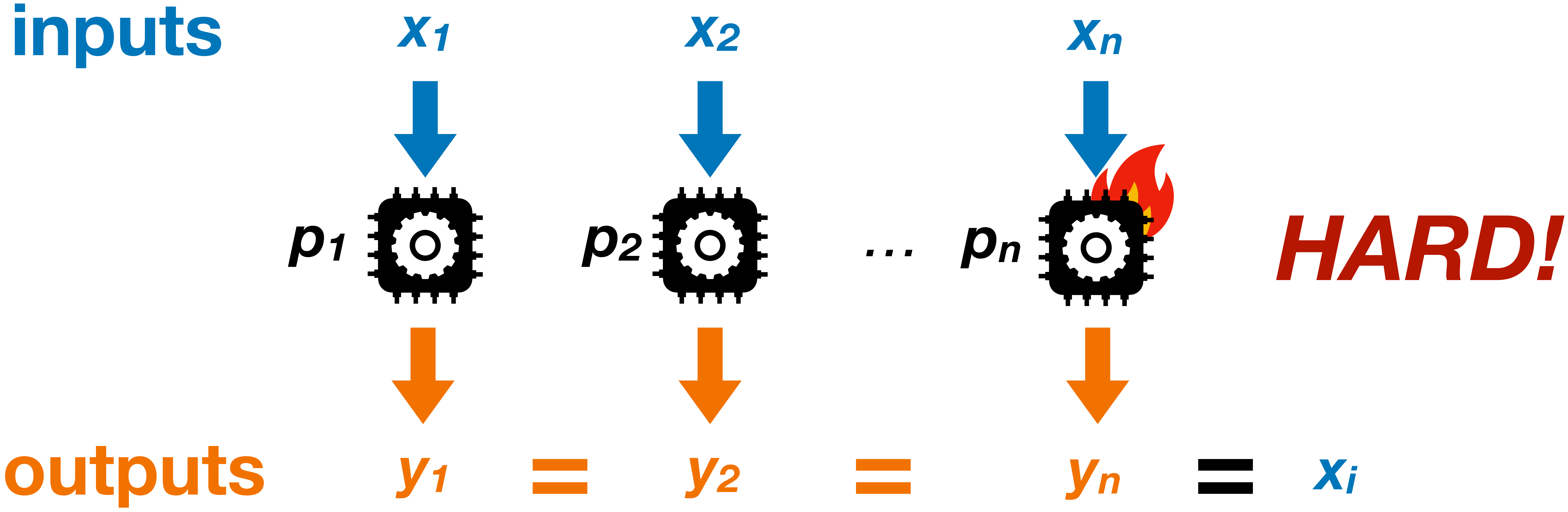
# Distributed agreement tasks



# Distributed agreement tasks:
consensus

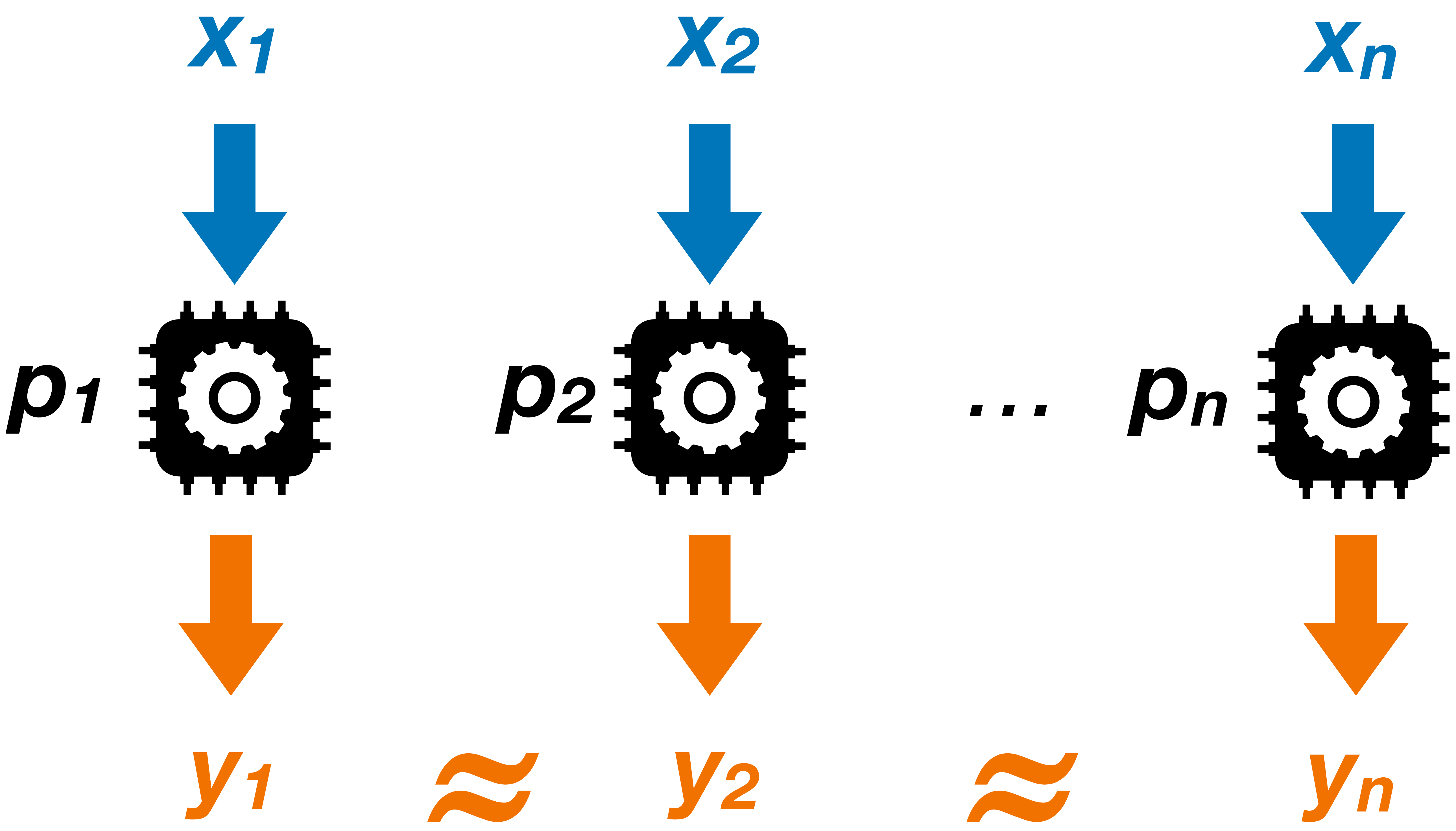


# Distributed agreement tasks:
consensus

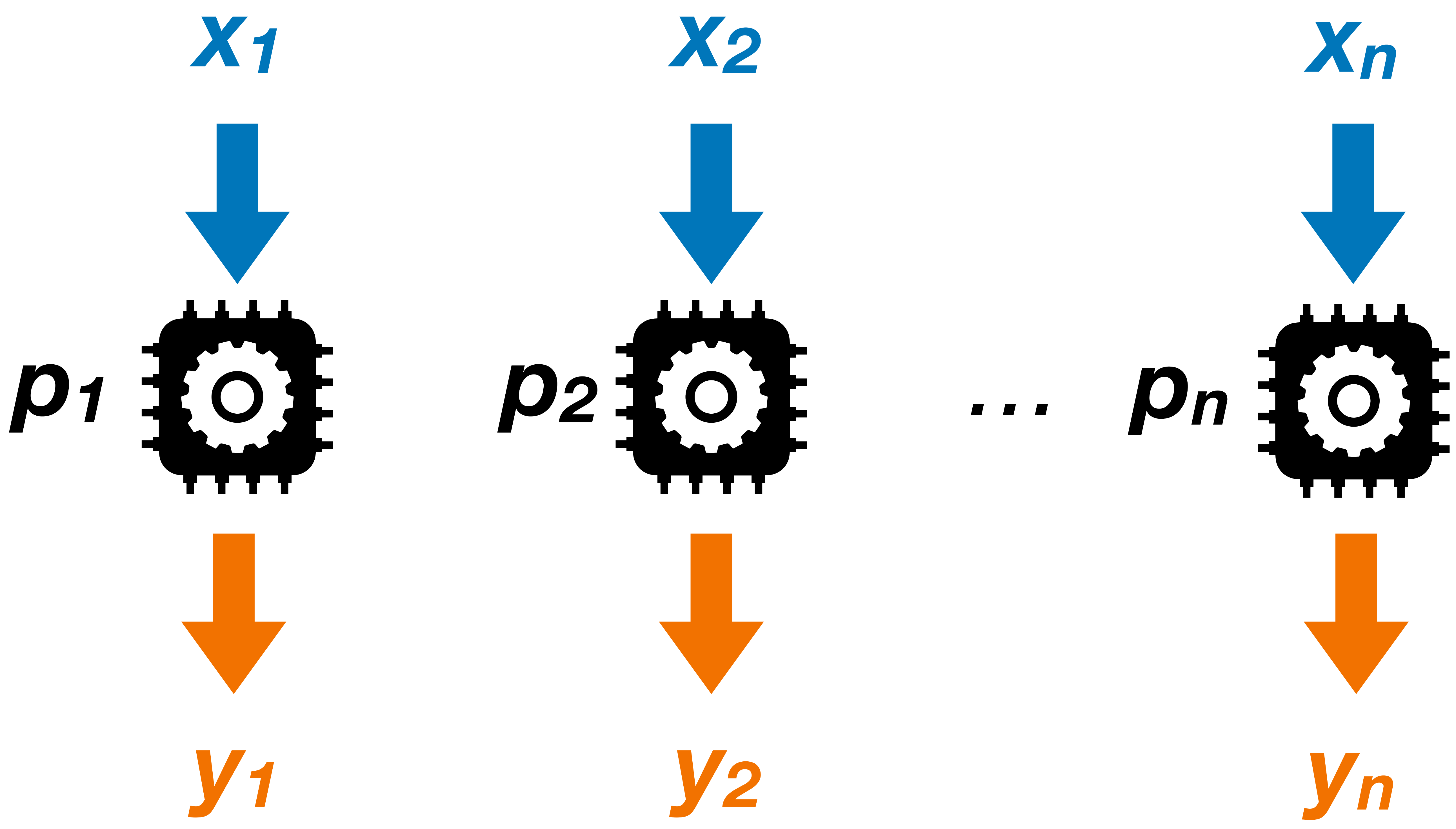



e.g., **Fischer, Lynch, Paterson (1985)**

# **Distributed agreement tasks:**
approximate agreement

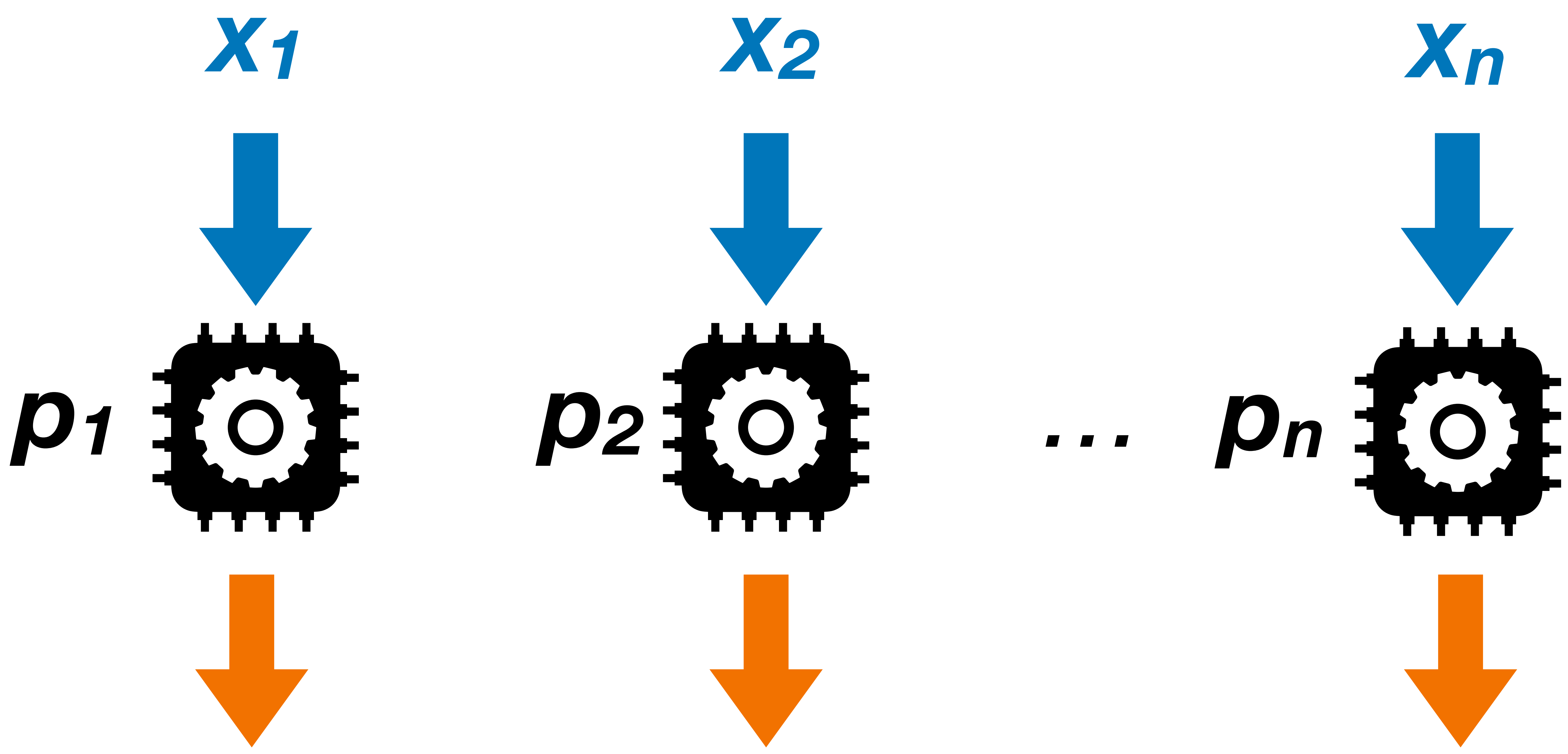


# Distributed agreement tasks:
approximate agreement over real numbers




distance( $y_i$ , $y_j$ ) $\leq \varepsilon$

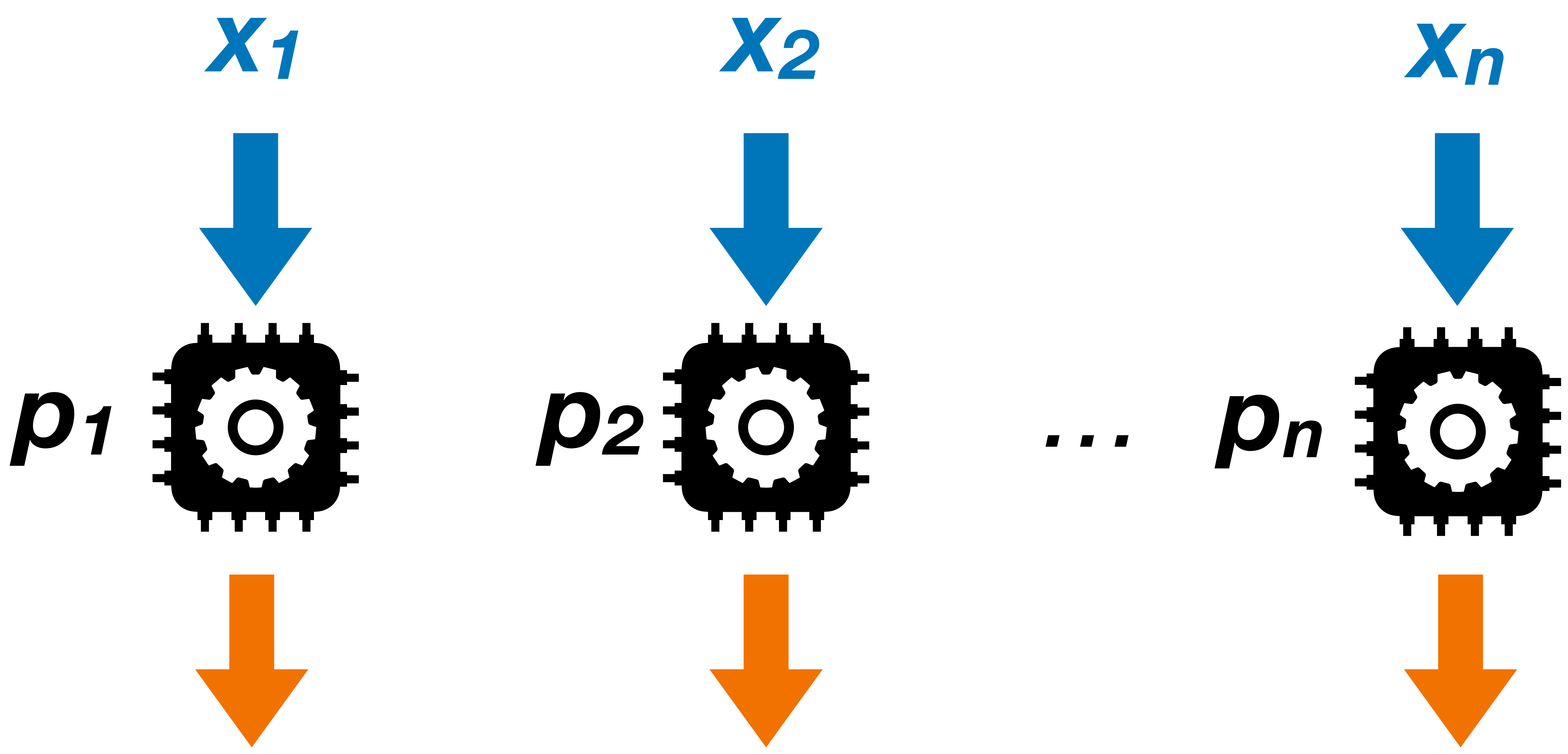
# **Distributed agreement tasks:**
## approximate agreement over real numbers




$$\min x_i \quad \leq \quad y_1 \qquad y_2 \qquad y_n \quad \leq \quad \max x_i$$

$$\mathrm{distance}(y_i, y_j) \leq \varepsilon$$

# **Distributed agreement tasks:**
## approximate agreement over real numbers




**EASY!**

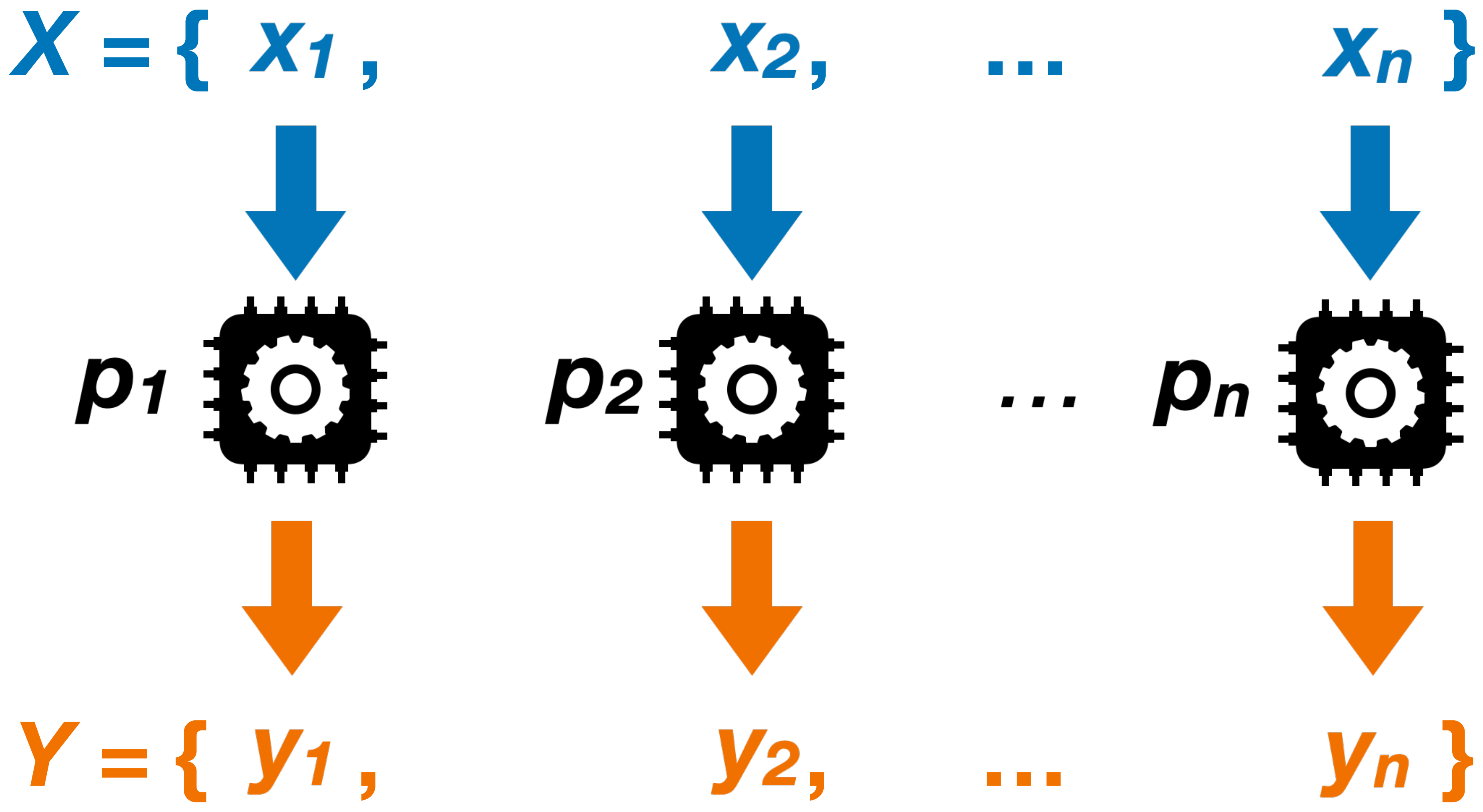
$$\min x_i \le y_1 \quad y_2 \quad \dots \quad y_n \le \max x_i$$

$$\text{distance}(y_i, y_j) \le \varepsilon$$

e.g., **Dolev, Lynch, Pinter, Stark, Weihl (1986)**

# Approximate agreement tasks

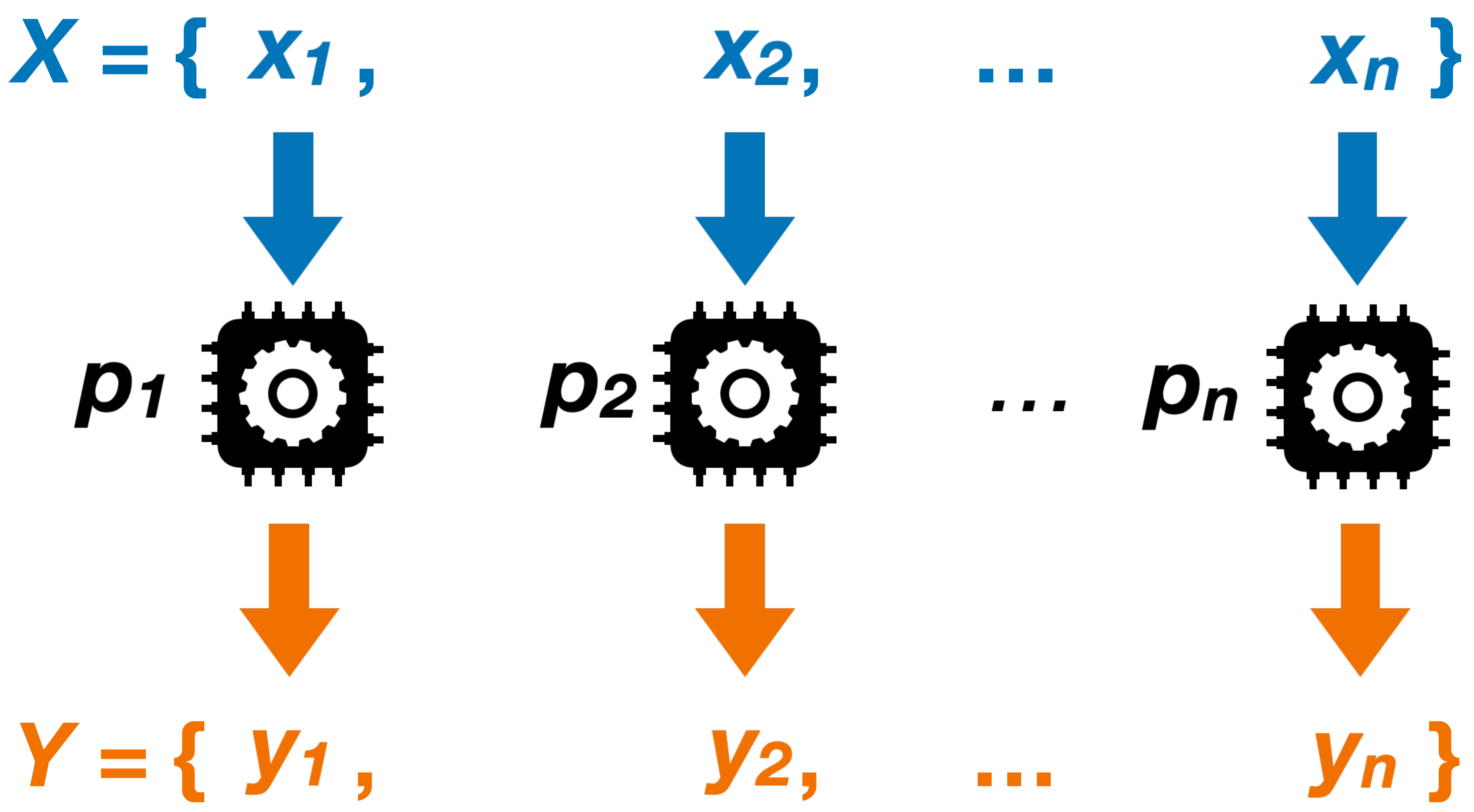
A fixed set $\boldsymbol{V}$ of values



# Approximate agreement tasks

A fixed set $\boldsymbol{V}$ of values

**Agreement:**
distance( $y_i, y_j$ ) $\leq \varepsilon$ for any $y_i$, $y_j$

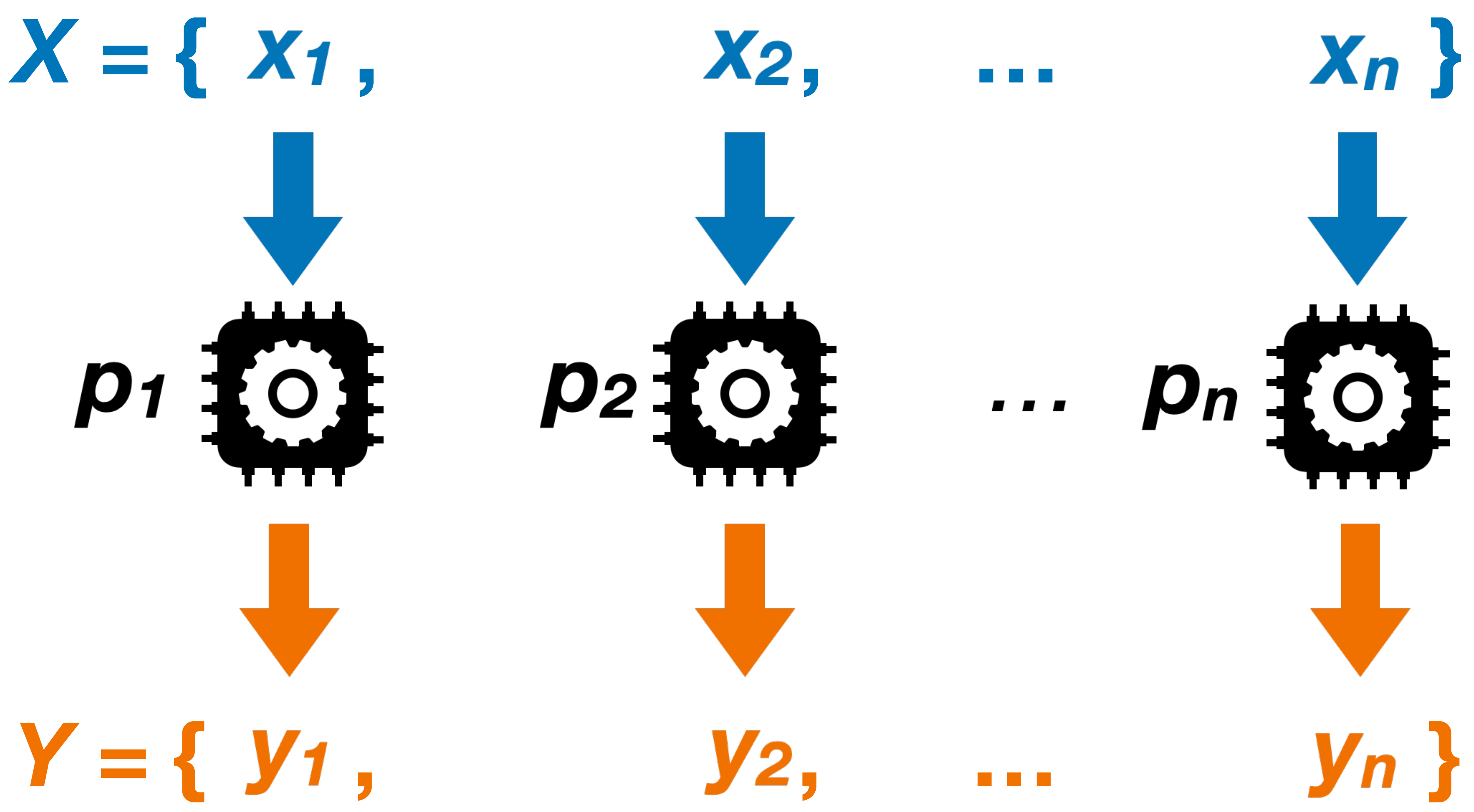


# Approximate agreement tasks

A fixed set $\boldsymbol{V}$ of values

**Agreement:**
distance( $y_i, y_j$ ) $\leq \varepsilon$ for any $y_i$, $y_j$

**Validity:**
$Y \subseteq \langle X \rangle$,
where $\langle \cdot \rangle$ is a closure operator on $\boldsymbol{V}$

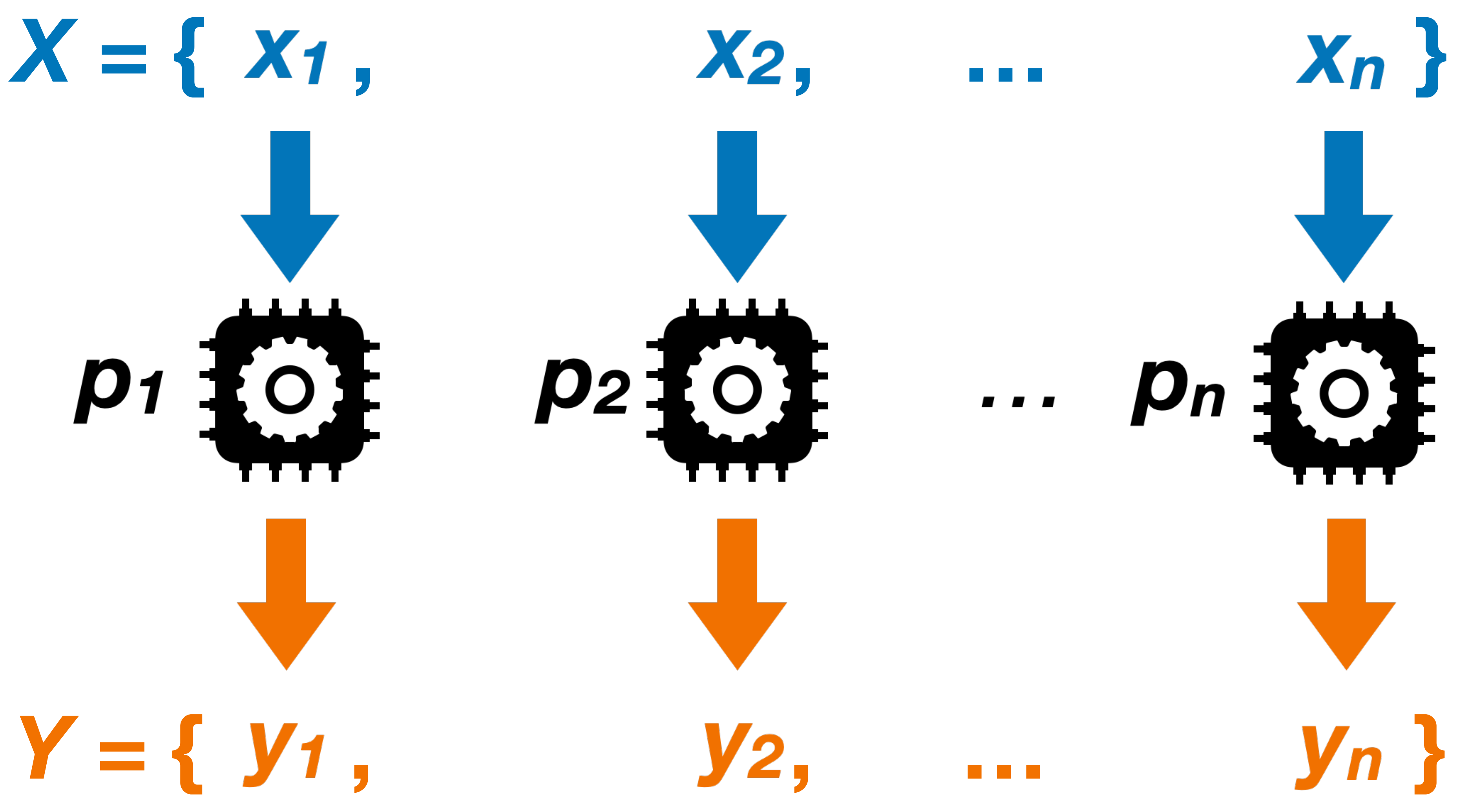


# Approximate agreement tasks

A fixed set ***V*** of values

**Agreement:**
distance( $y_i, y_j$ ) $\leq \varepsilon$ for any $y_i$, $y_j$

**Validity:**
$Y \subseteq \langle X \rangle$,
where **< · >** is a closure operator on ***V***




= "**outputs** are close to one another and reside in some **closure** of the **input** values"

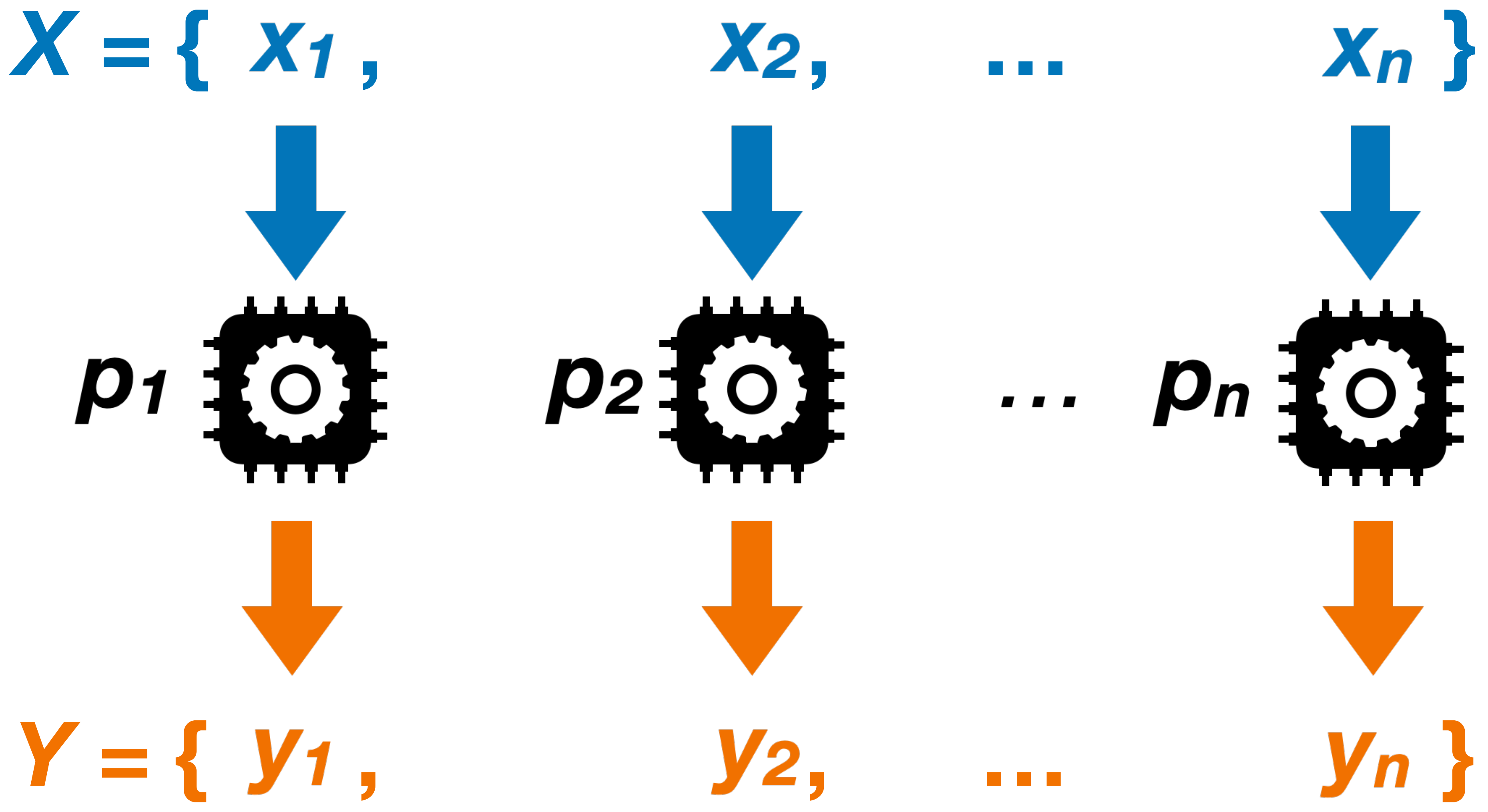
# Approximate agreement tasks

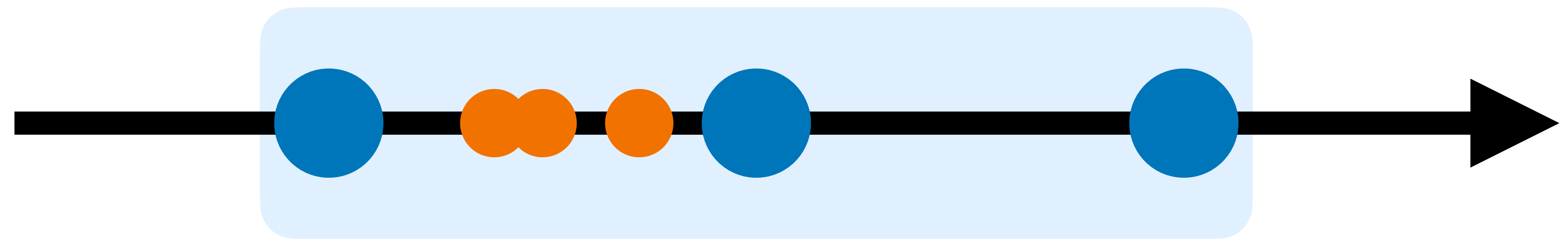
A fixed set ***V*** of values

**Agreement:**
distance( $y_i, y_j$ ) $\leq \varepsilon$ for any $y_i$, $y_j$

**Validity:**
$Y \subseteq \langle X \rangle$,
where $\langle \cdot \rangle$ is a closure operator on ***V***

= “**outputs** are close to one another and reside in some **closure** of the **input** values”



e.g., **Dolev, Lynch, Pinter, Stark, Weihl (1986)**
**Abraham, Amit, Dolev (2004)**
approximate agreement over reals

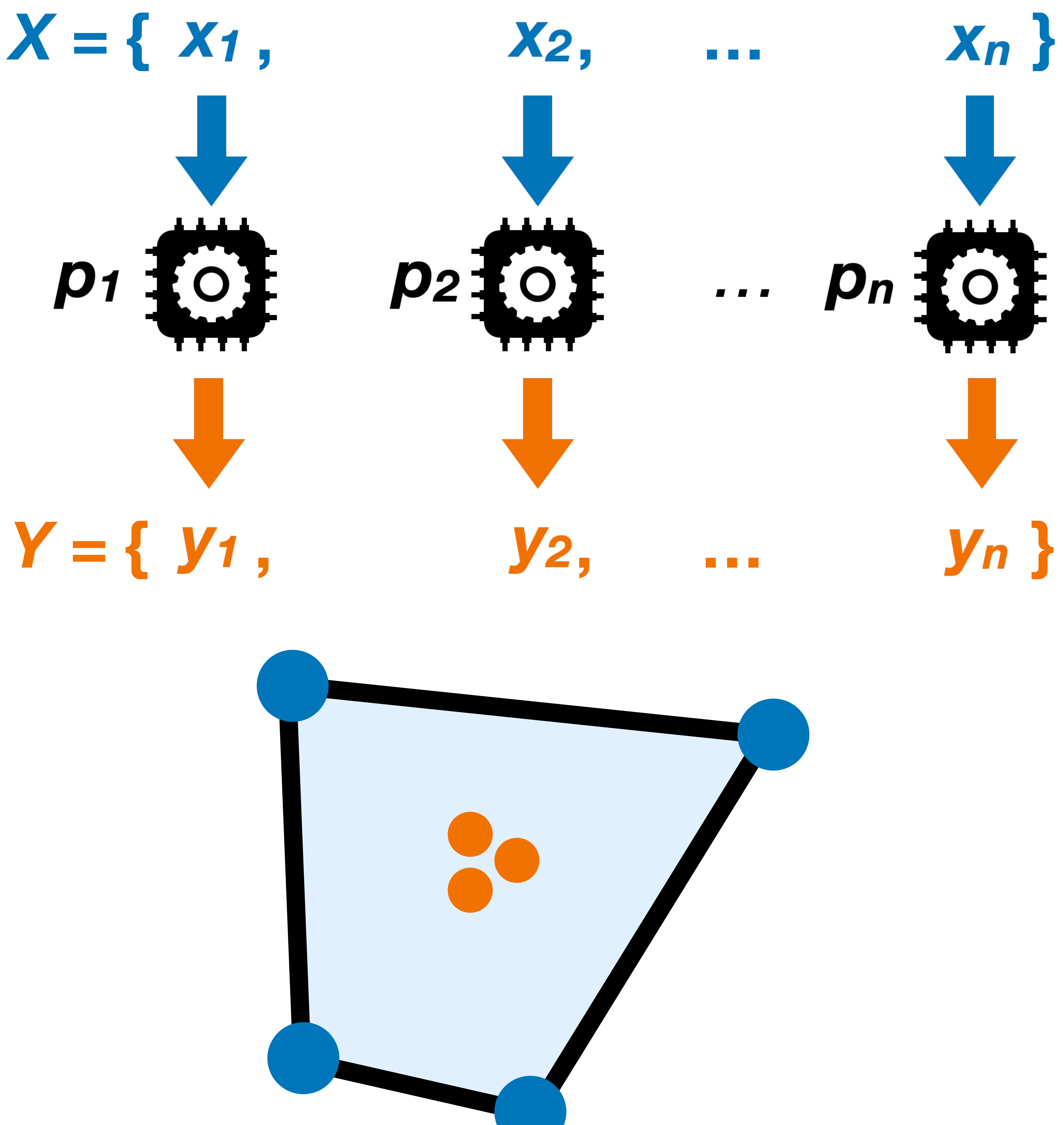
# Approximate agreement tasks

A fixed set $\boldsymbol{V}$ of values

**Agreement:**
distance( $y_i, y_j$ ) $\leq \varepsilon$ for any $y_i$, $y_j$

**Validity:**
$Y \subseteq \langle X \rangle$,
where $\langle \cdot \rangle$ is a closure operator on $\boldsymbol{V}$

= "**outputs** are close to one another and reside in some **closure** of the **input** values"




e.g., **Mendes, Herlihy, Vaidya, Garg (2015)**
multidimensional approximate agreement

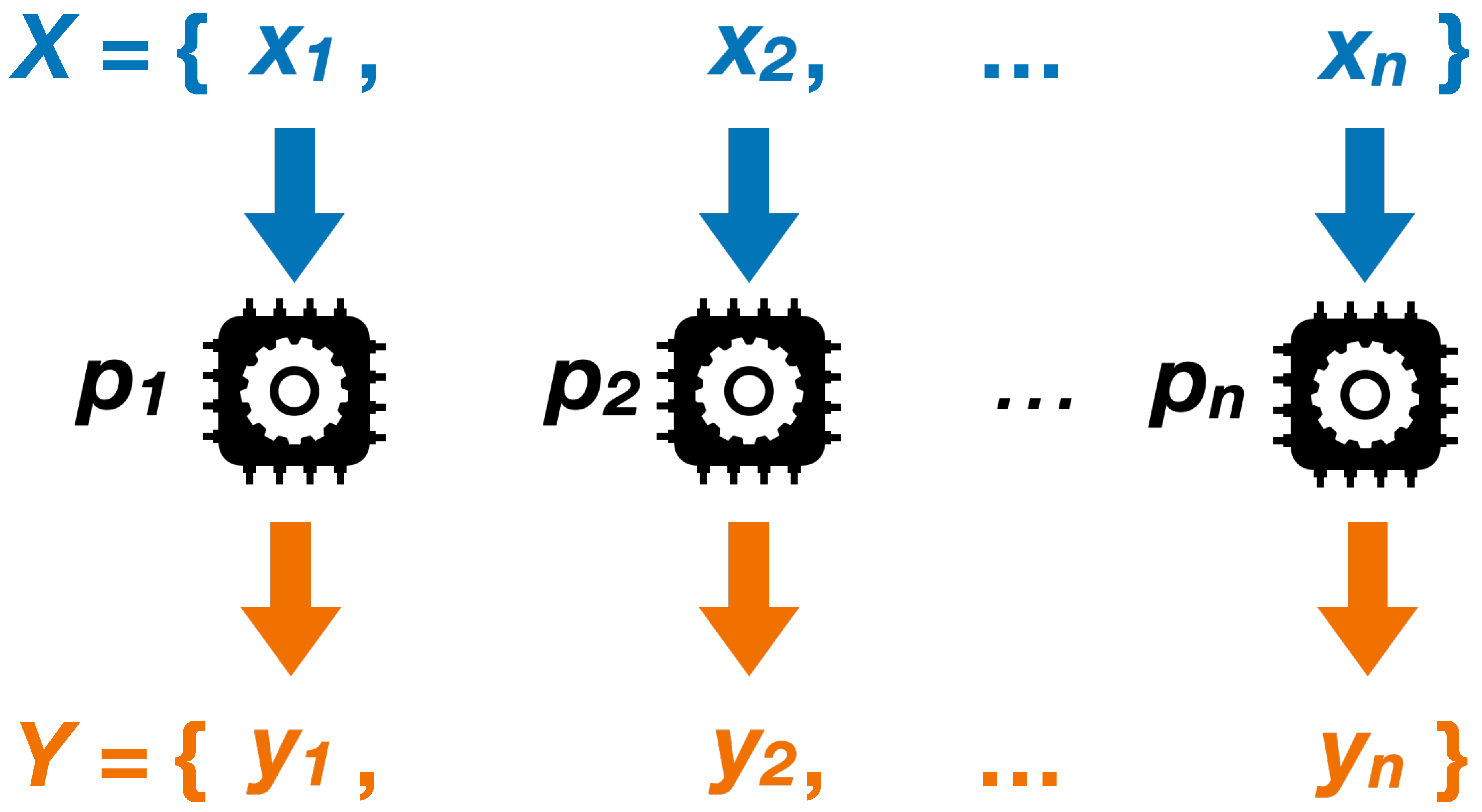
# Approximate agreement tasks

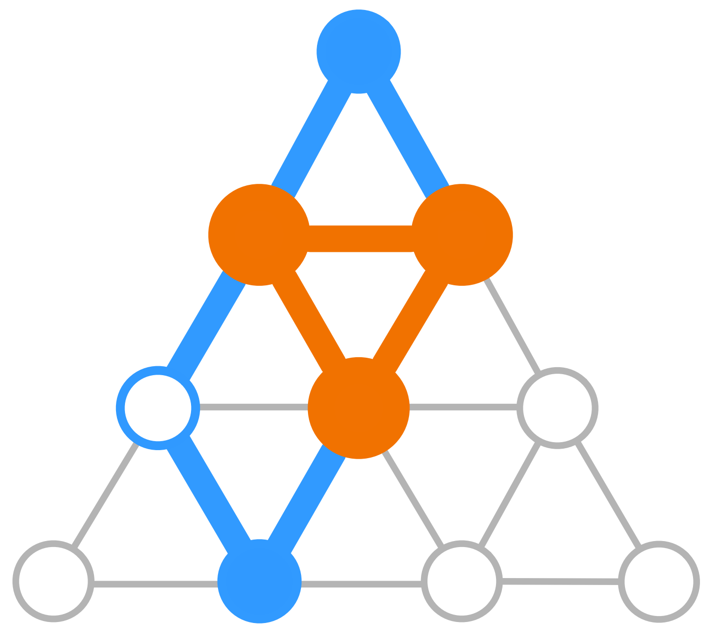
A fixed set ***V*** of values

**Agreement:**
distance( $y_i, y_j$ ) $\leq \varepsilon$ for any $y_i$, $y_j$

**Validity:**
$Y \subseteq \langle X \rangle$,
where $\langle \cdot \rangle$ is a closure operator on ***V***

= "**outputs** are close to one another and reside in some **closure** of the **input** values"

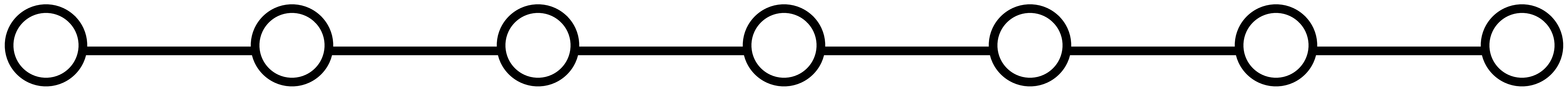


e.g. **Nowak and Rybicki (DISC 2019)**
approximate agreement on graphs and lattices

# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

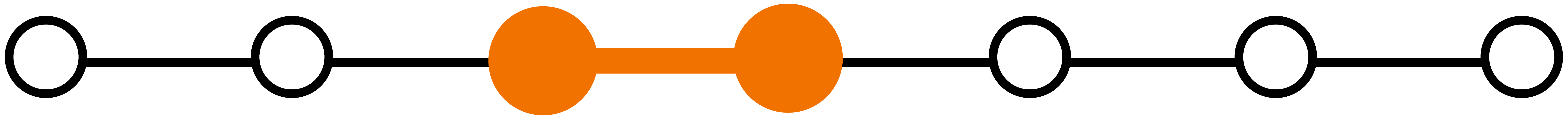


***G***

# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***



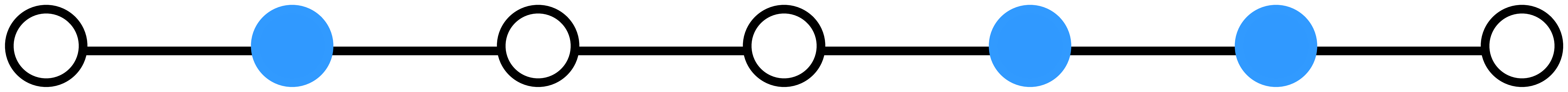
***G***

# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**



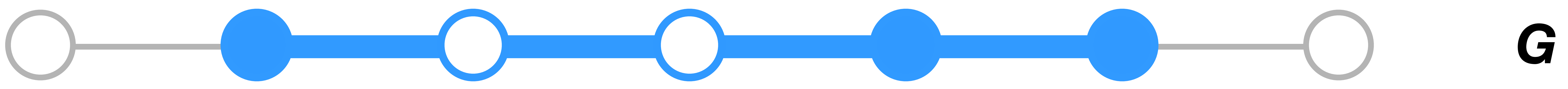
***G***

# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**




“shortest path convex hull”

# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**

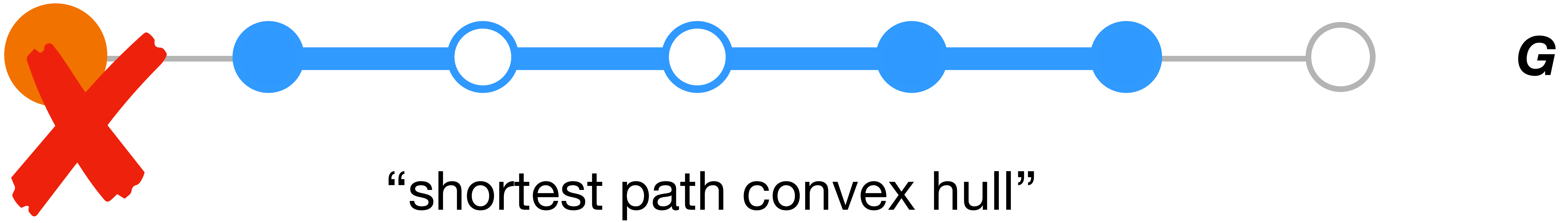

“shortest path convex hull”

# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**



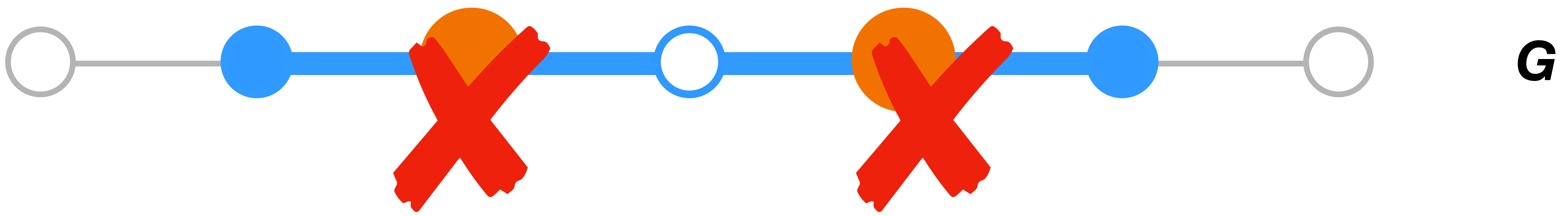

"shortest path convex hull"

# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**

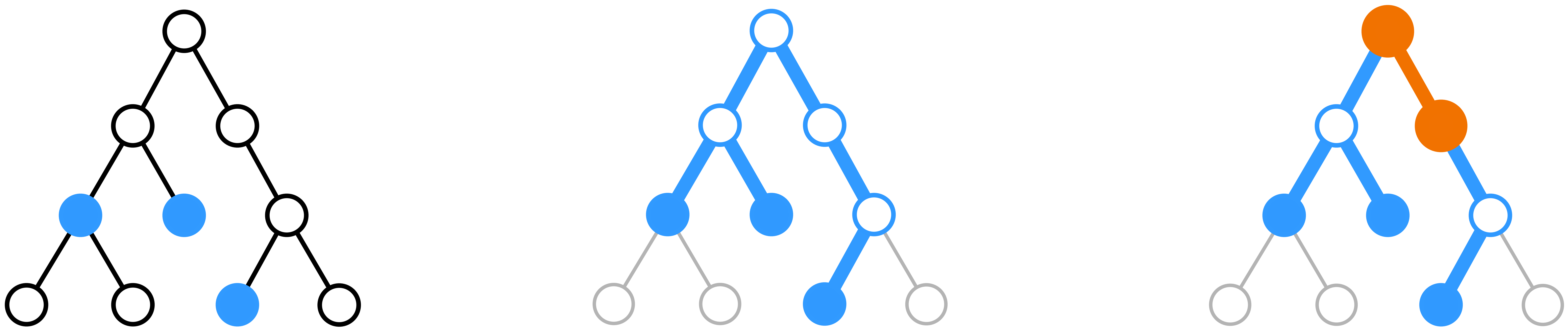


# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**

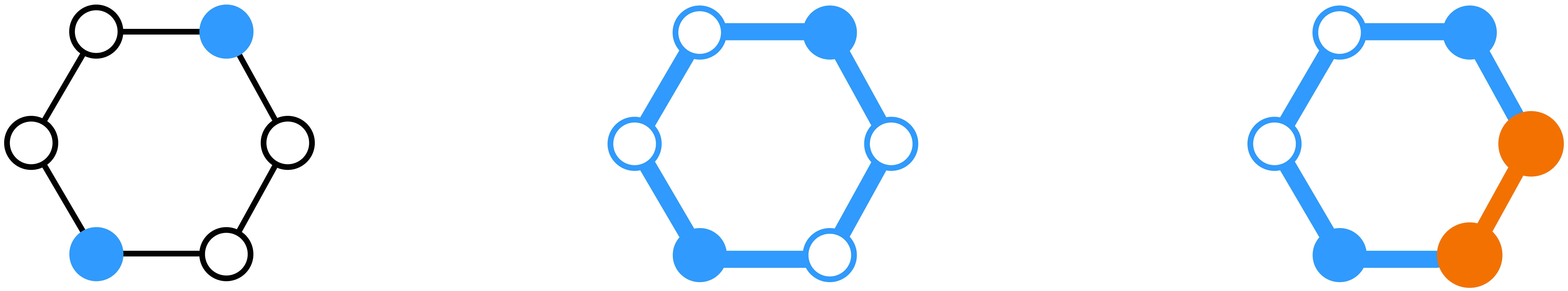


# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**

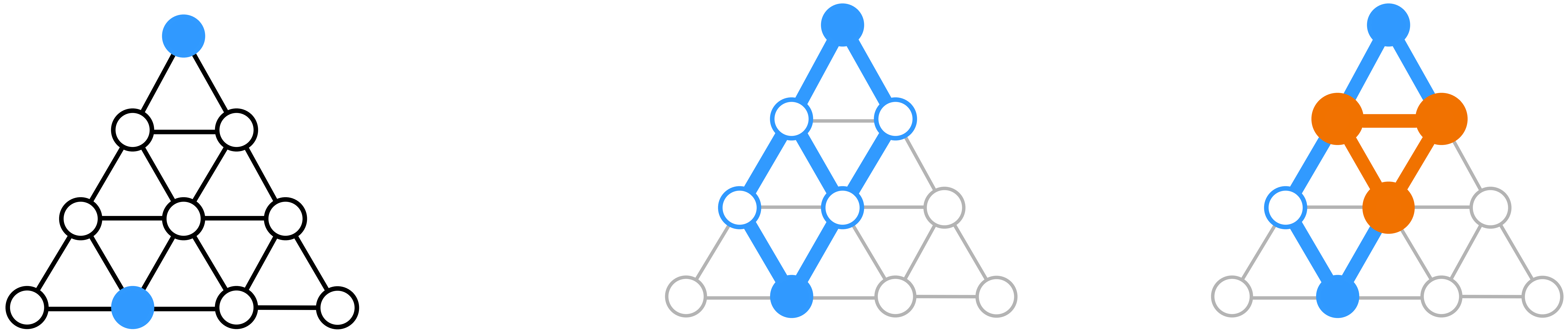


# Approximate agreement on graphs

**Inputs** and **outputs** are vertices of a fixed graph ***G***

**Agreement:**
**outputs** form a clique of ***G***

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**



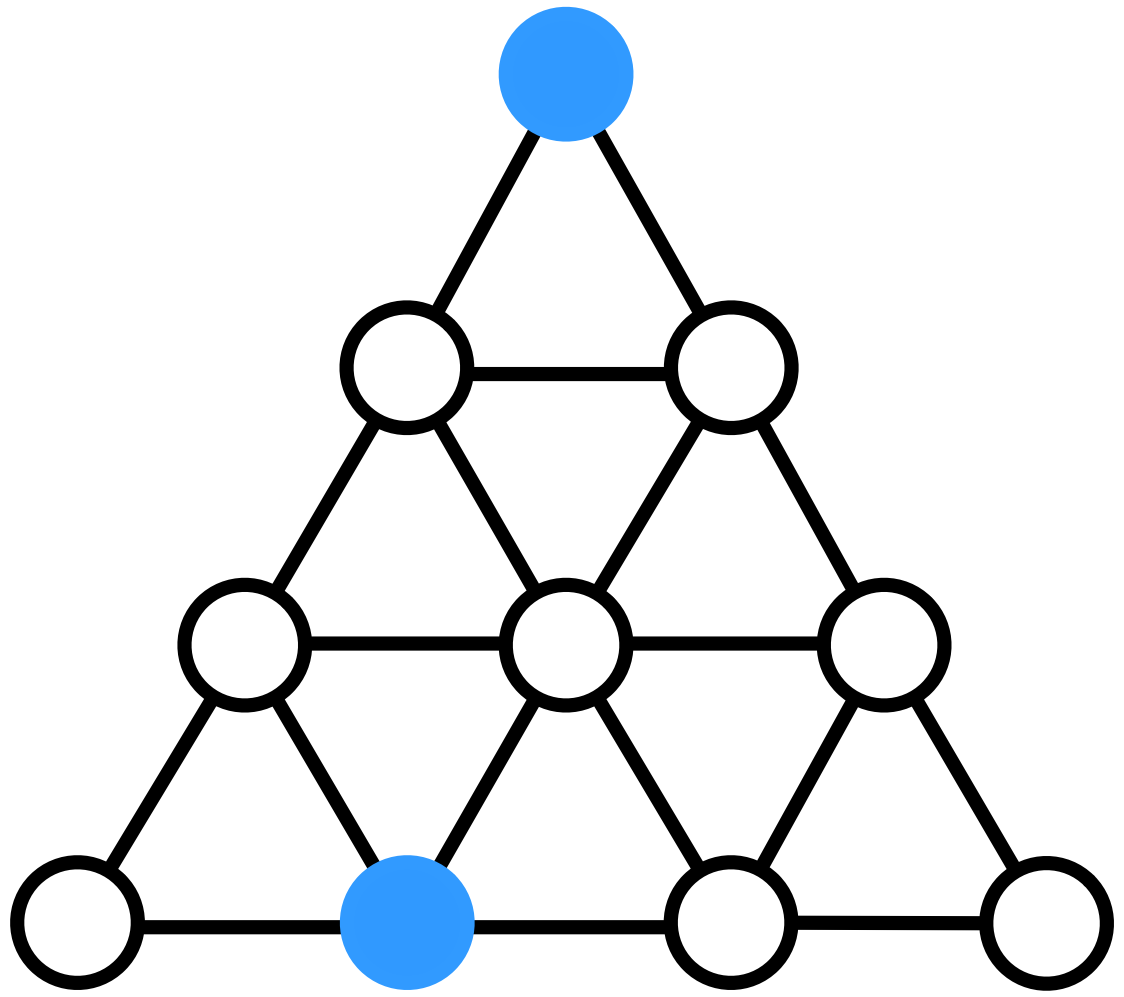
# **Other validity conditions:** minimal path validity

Each output on a ***minimal path*** between **inputs**
(every path that is an induced path in ***G***)

**Nowak and Rybicki (DISC 2019)**
Message-passing + arbitrary faults

# **Other validity conditions:**
# minimal path validity

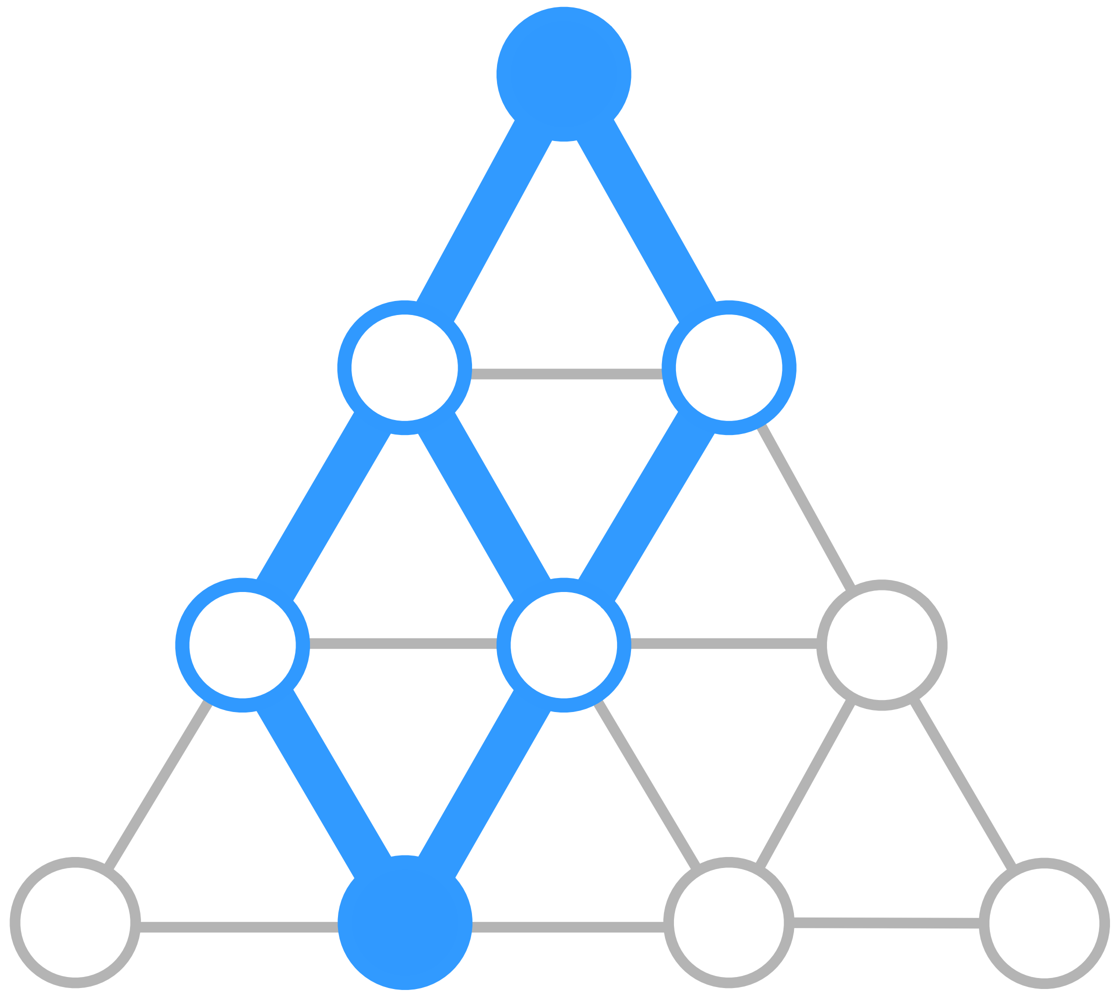
Each output on a ***minimal path*** between **inputs**
(every path that is an induced path in ***G***)



**Nowak and Rybicki (DISC 2019)**
Message-passing + arbitrary faults

# **Other validity conditions:**
# minimal path validity

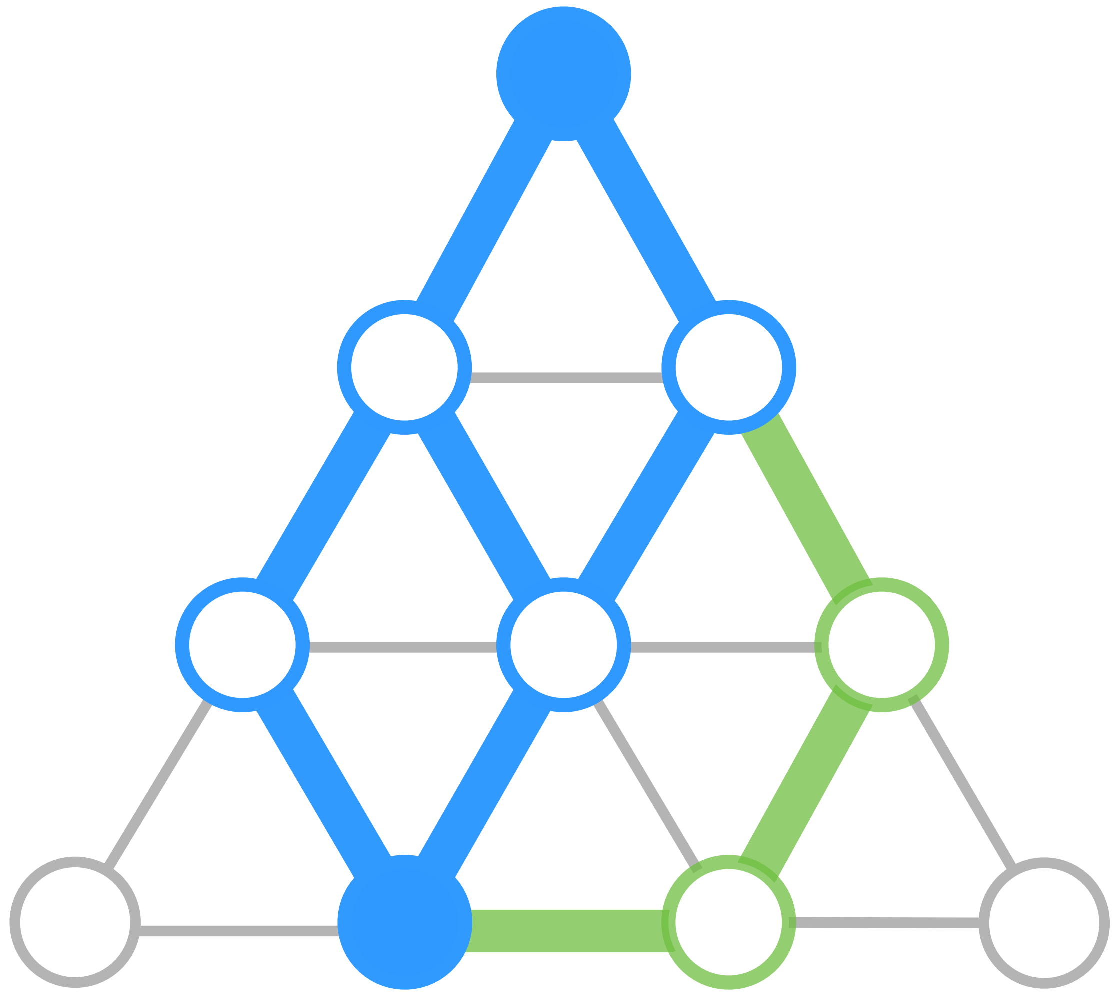
Each output on a ***minimal path*** between **inputs**
(every path that is an induced path in ***G***)



**Nowak and Rybicki (DISC 2019)**
Message-passing + arbitrary faults

# **Other validity conditions:**
# minimal path validity

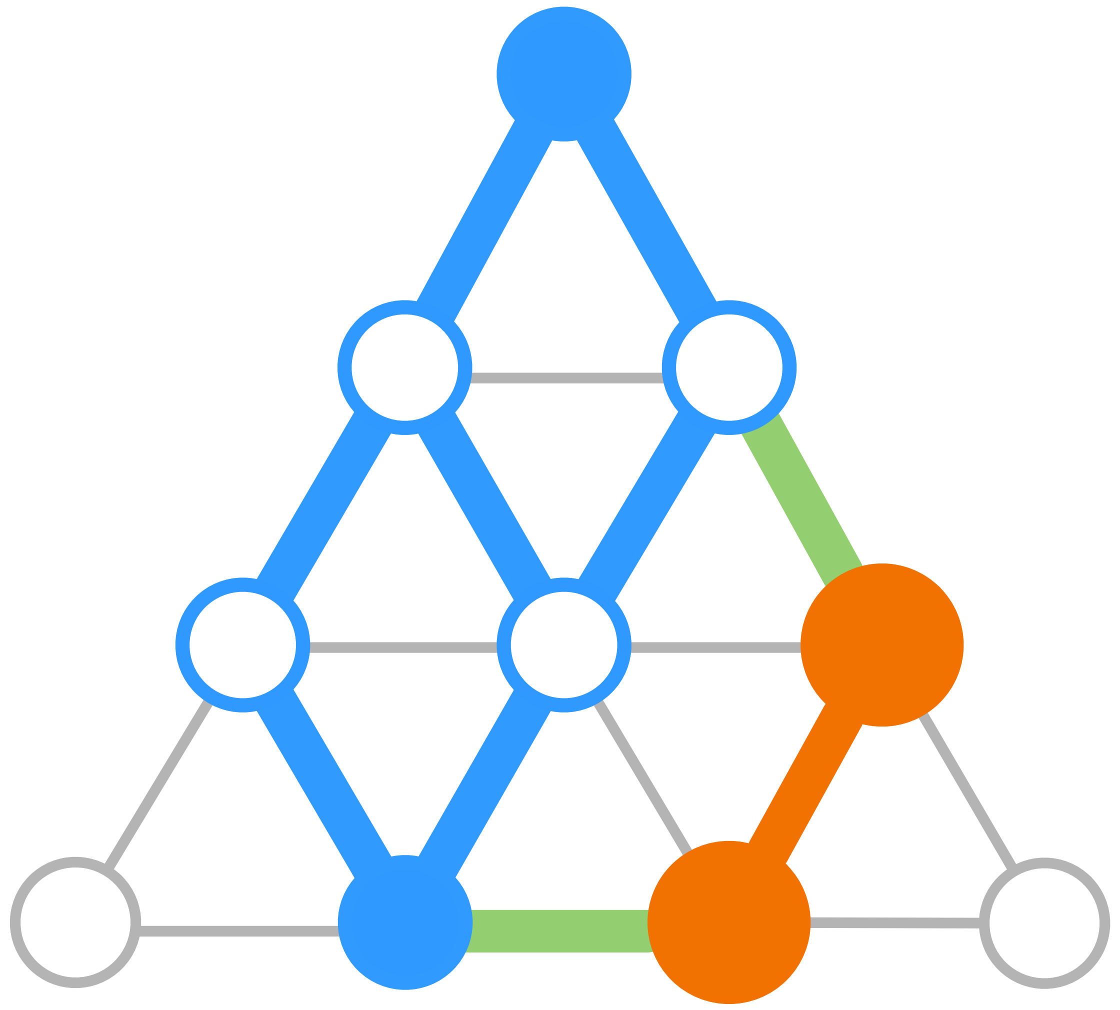
Each output on a ***minimal path*** between **inputs**
(every path that is an induced path in ***G***)



**Nowak and Rybicki (DISC 2019)**
Message-passing + arbitrary faults

# **Other validity conditions:**
# minimal path validity

Each output on a ***minimal path*** between **inputs**
(every path that is an induced path in ***G***)



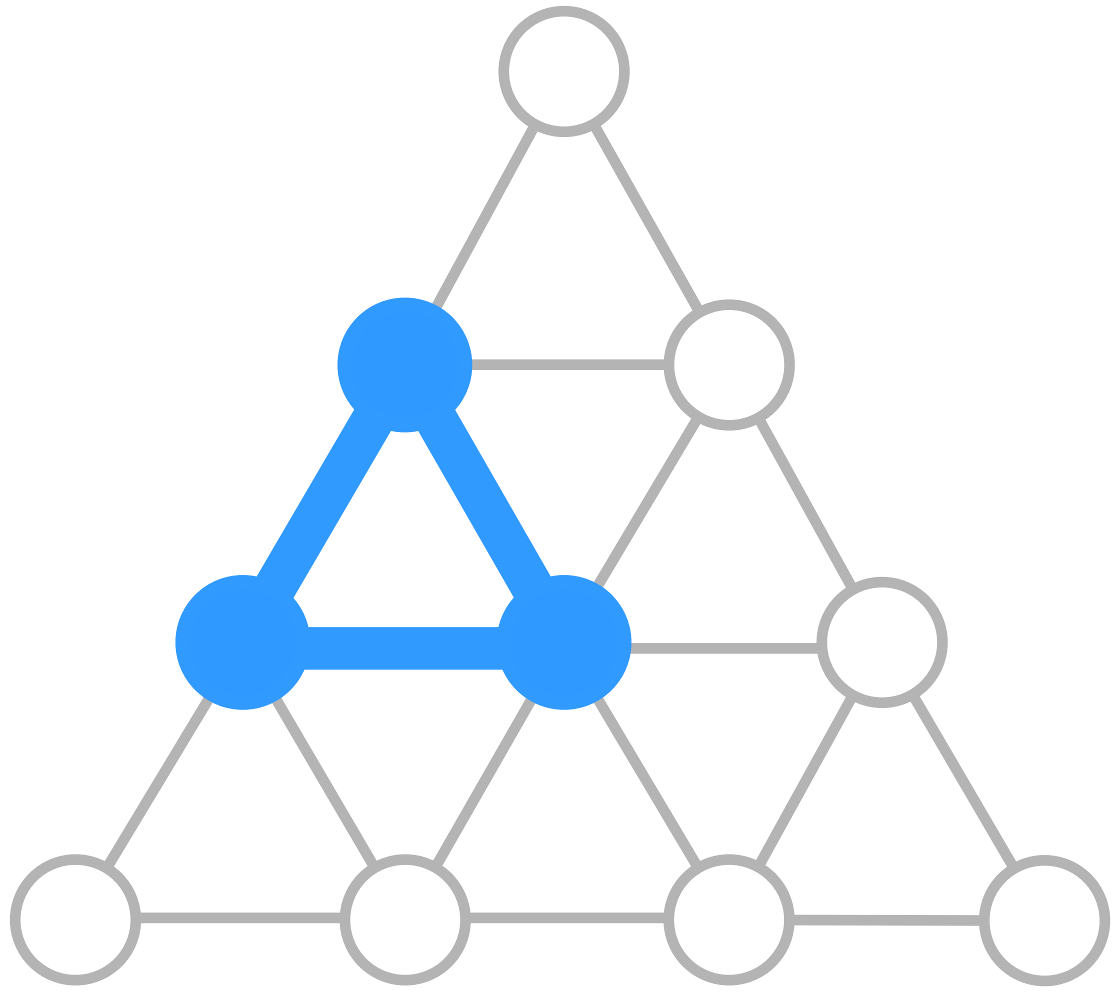
**Nowak and Rybicki (DISC 2019)**
Message-passing + arbitrary faults

# **Other validity conditions:**
clique validity

**Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum (Distributed Computing 2019)**
robots in look-compute-move models

# Other validity conditions:
clique validity

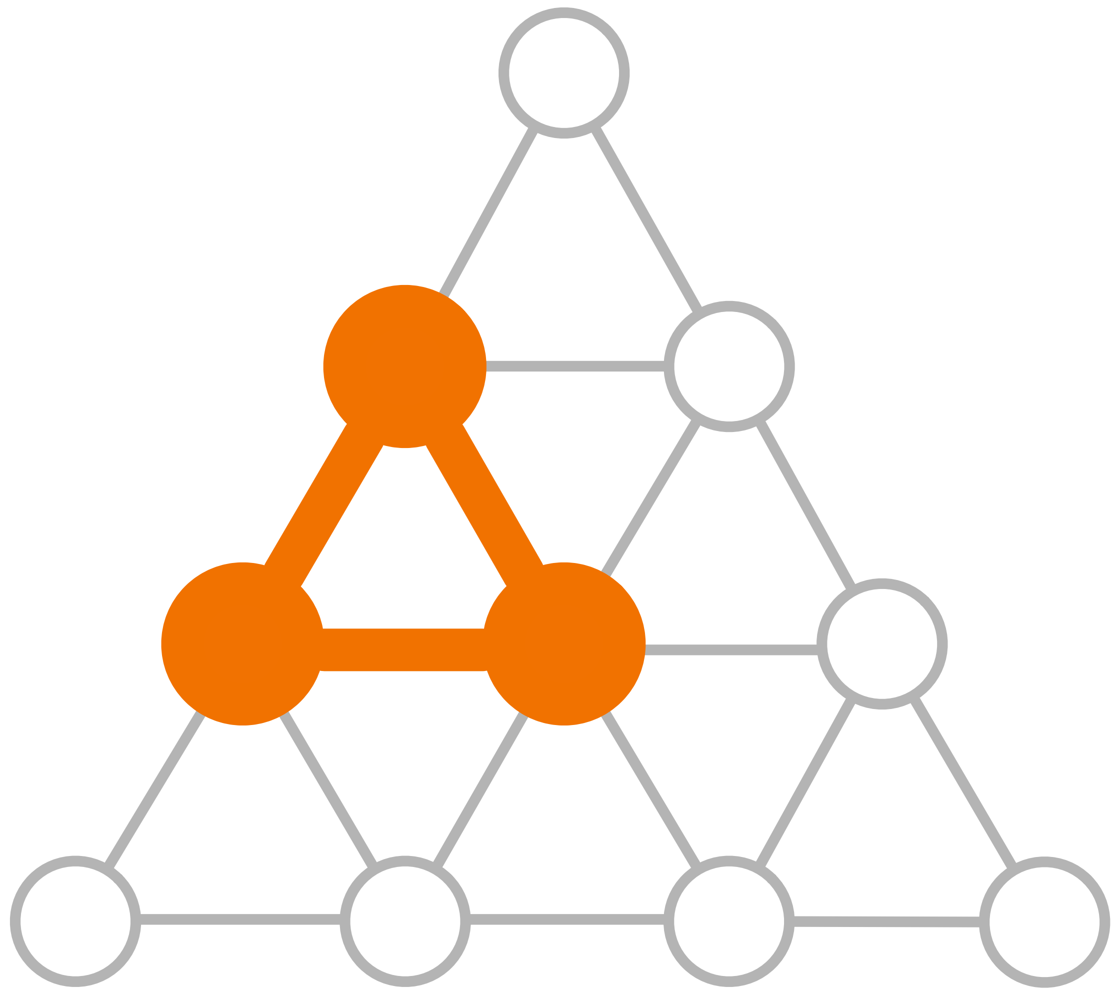
If set $X$ of **inputs** forms a clique in $G$, then $Y \subseteq X$
(otherwise set $Y$ of **outputs** can be any clique)



**Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum (Distributed Computing 2019)**
robots in look-compute-move models

# Other validity conditions:
clique validity

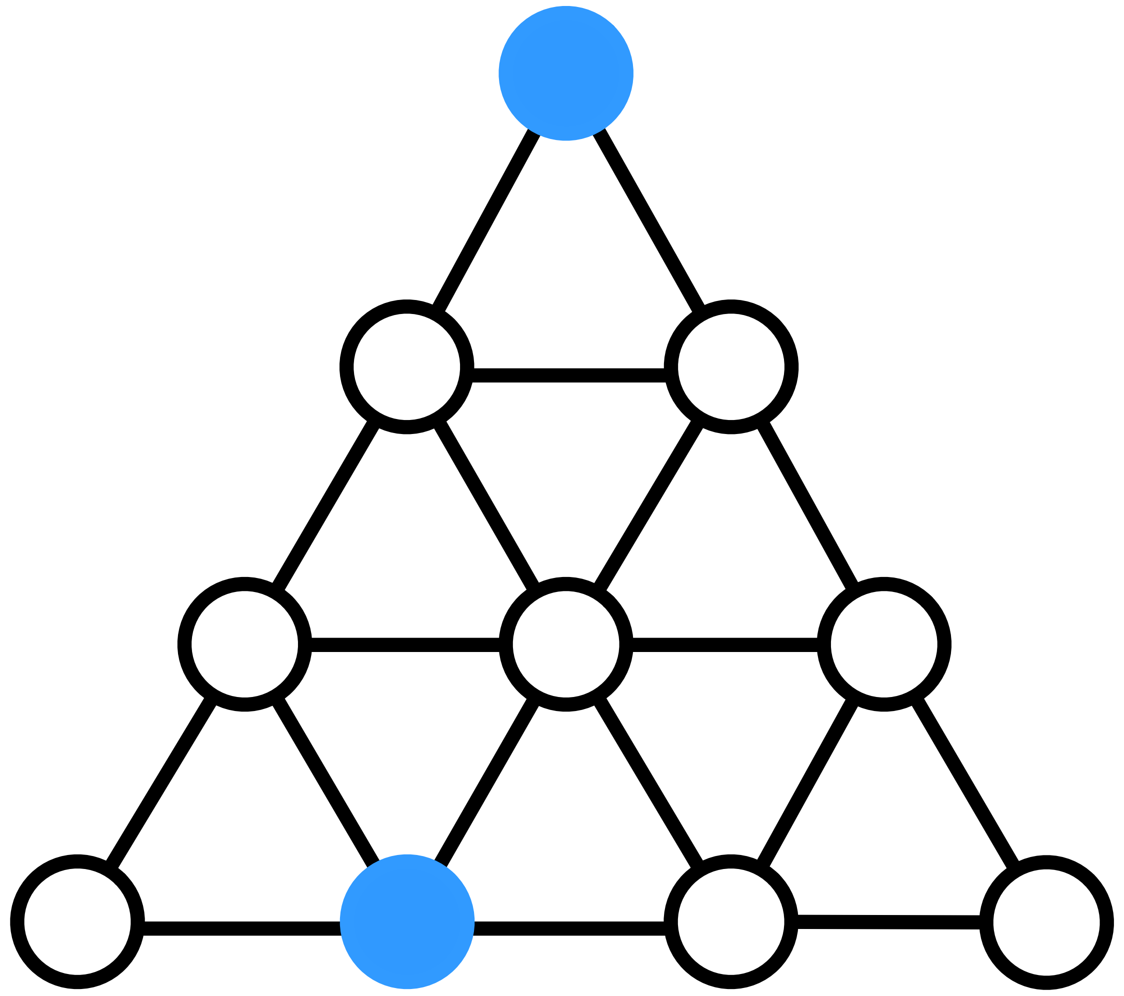
If set ***X*** of **inputs** forms a clique in ***G***, then ***Y*** $\subseteq$ ***X***
(otherwise set ***Y*** of **outputs** can be any clique)



**Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum (Distributed Computing 2019)**
robots in look-compute-move models

# **Other validity conditions:** clique validity

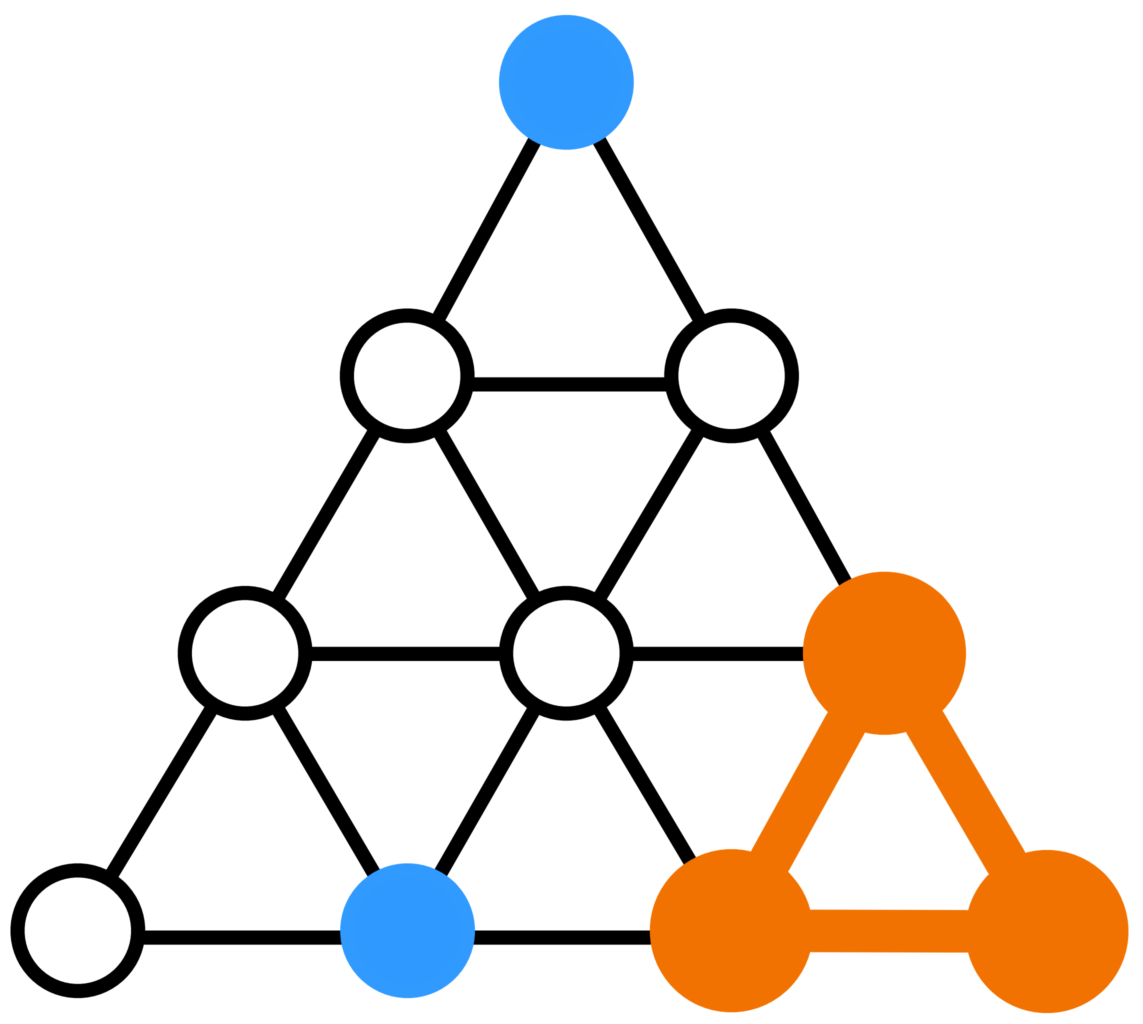
If set $X$ of **inputs** forms a clique in $G$, then $Y \subseteq X$
(otherwise set $Y$ of **outputs** can be any clique)



**Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum (Distributed Computing 2019)**
robots in look-compute-move models

# **Other validity conditions:** clique validity

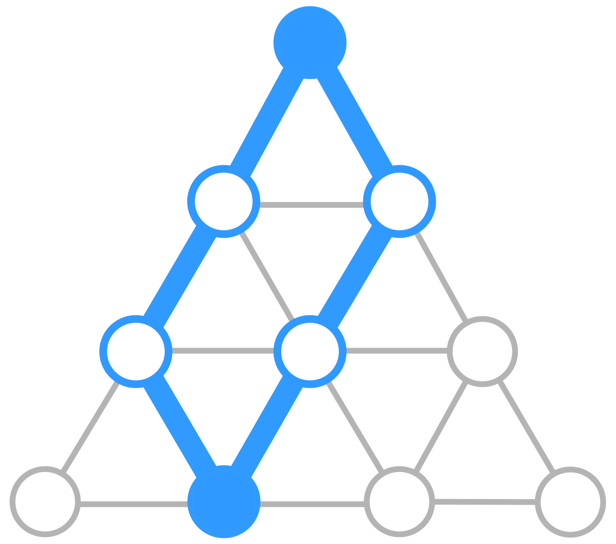
If set ***X*** of **inputs** forms a clique in ***G***, then ***Y*** ⊆ ***X***
(otherwise set ***Y*** of **outputs** can be any clique)



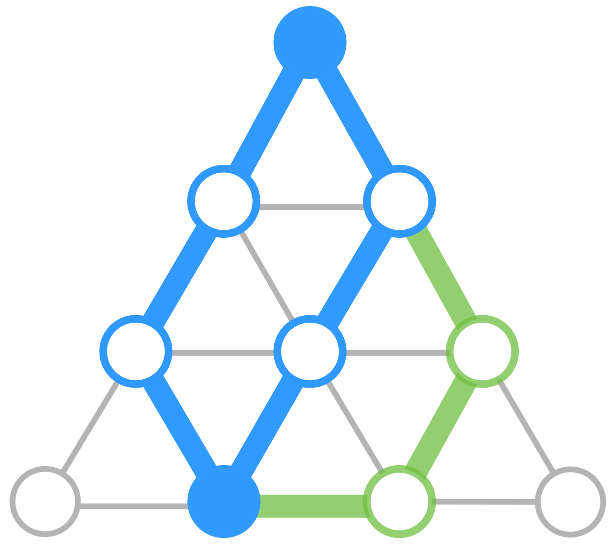
**Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum (Distributed Computing 2019)**
robots in look-compute-move models

# Comparison of validity conditions

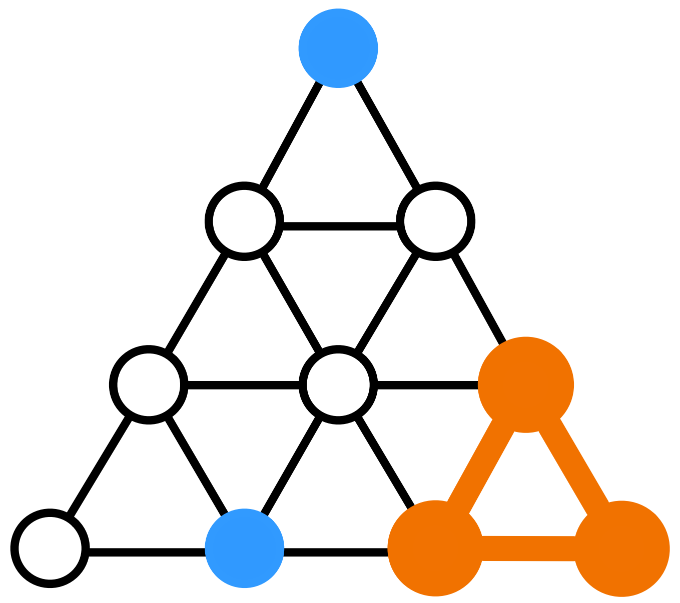
**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**



**Minimal path validity:**
Each **output** on a *minimal path* between **inputs**



**Clique validity:**
If set $X$ of **inputs** forms a clique in $G$, then $Y \subseteq X$



# Comparison of validity conditions

**Shortest path validity:**
each **output** on a *shortest path* between two **inputs**

**Nice for upper bounds!**

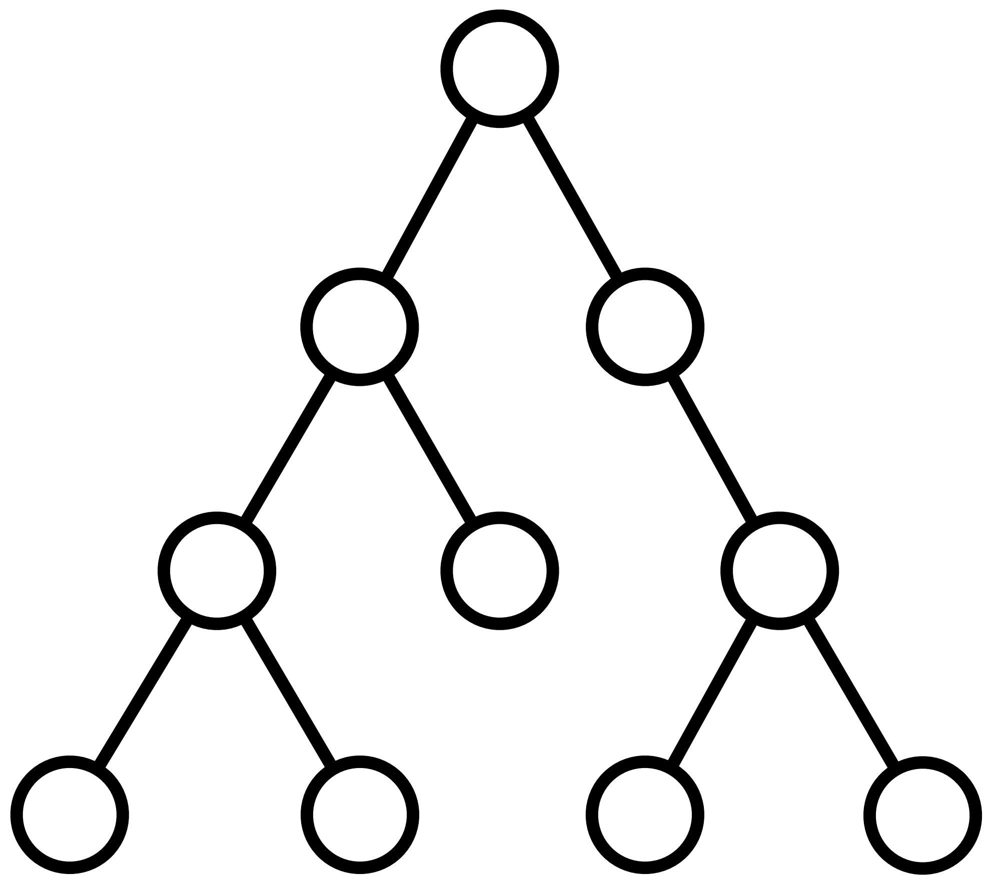
**Minimal path validity:**
Each **output** on a *minimal path* between **inputs**

**Clique validity:**
If set ***X*** of **inputs** forms a clique in ***G***, then $Y \subseteq X$

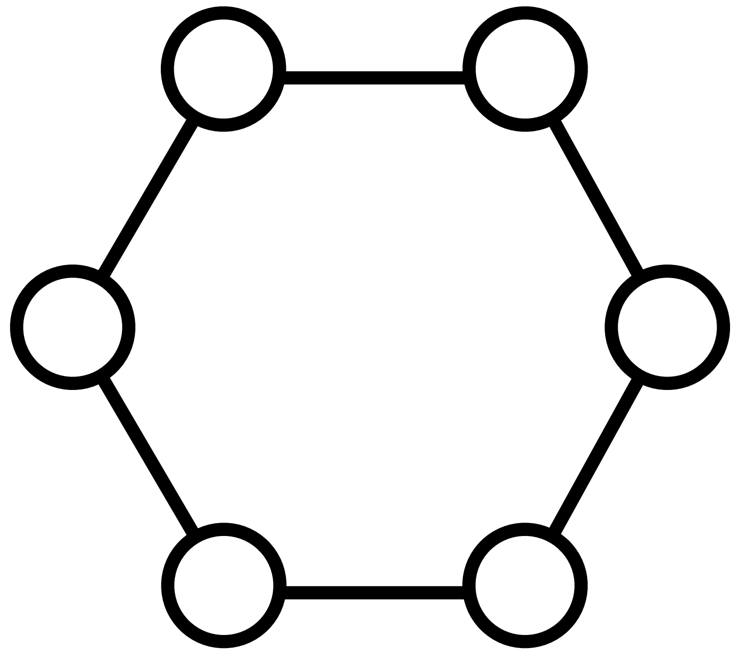
**Nice for lower bounds!**

Some other validity conditions also exist:
see e.g., Alcántara et al. (2019)

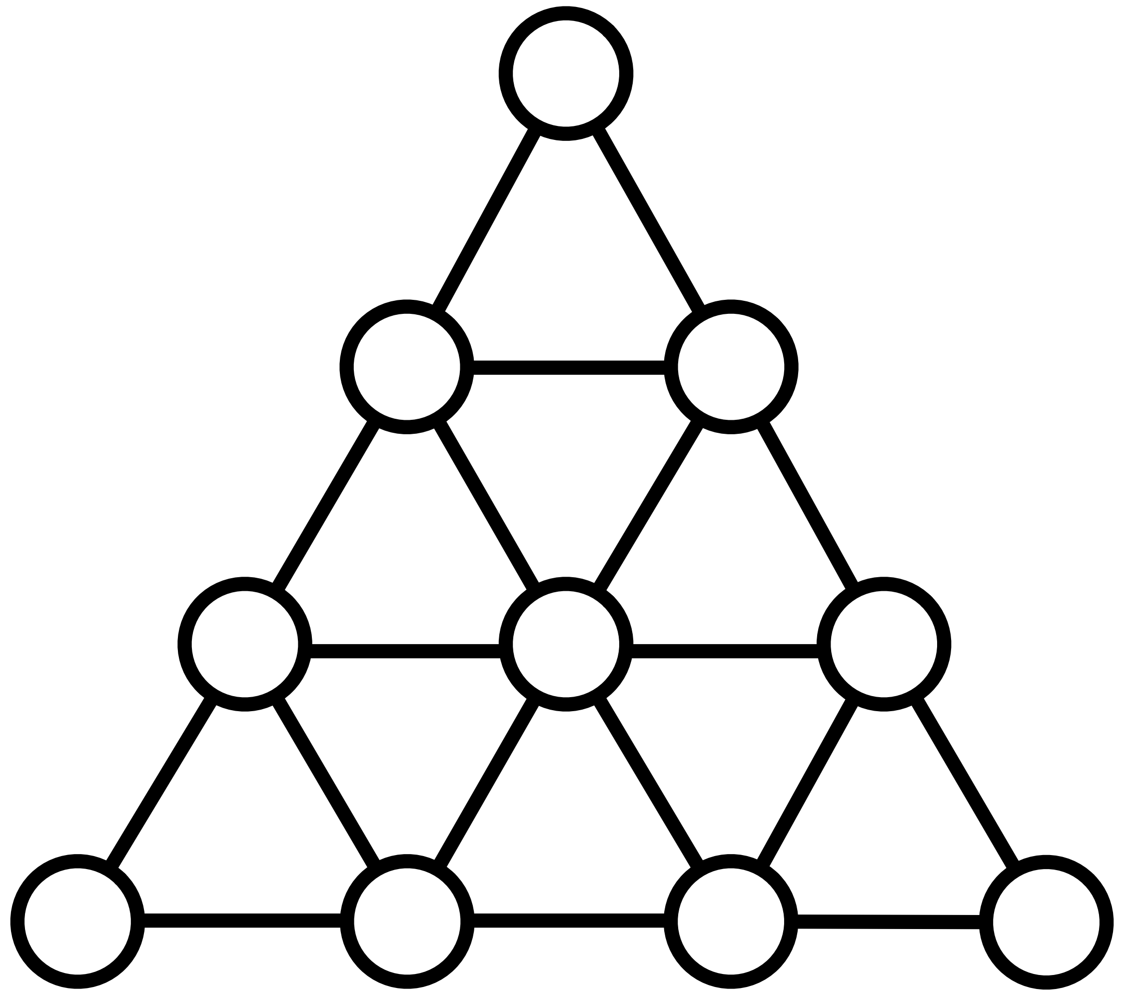
# On what graphs is approximate agreement wait-free solvable?



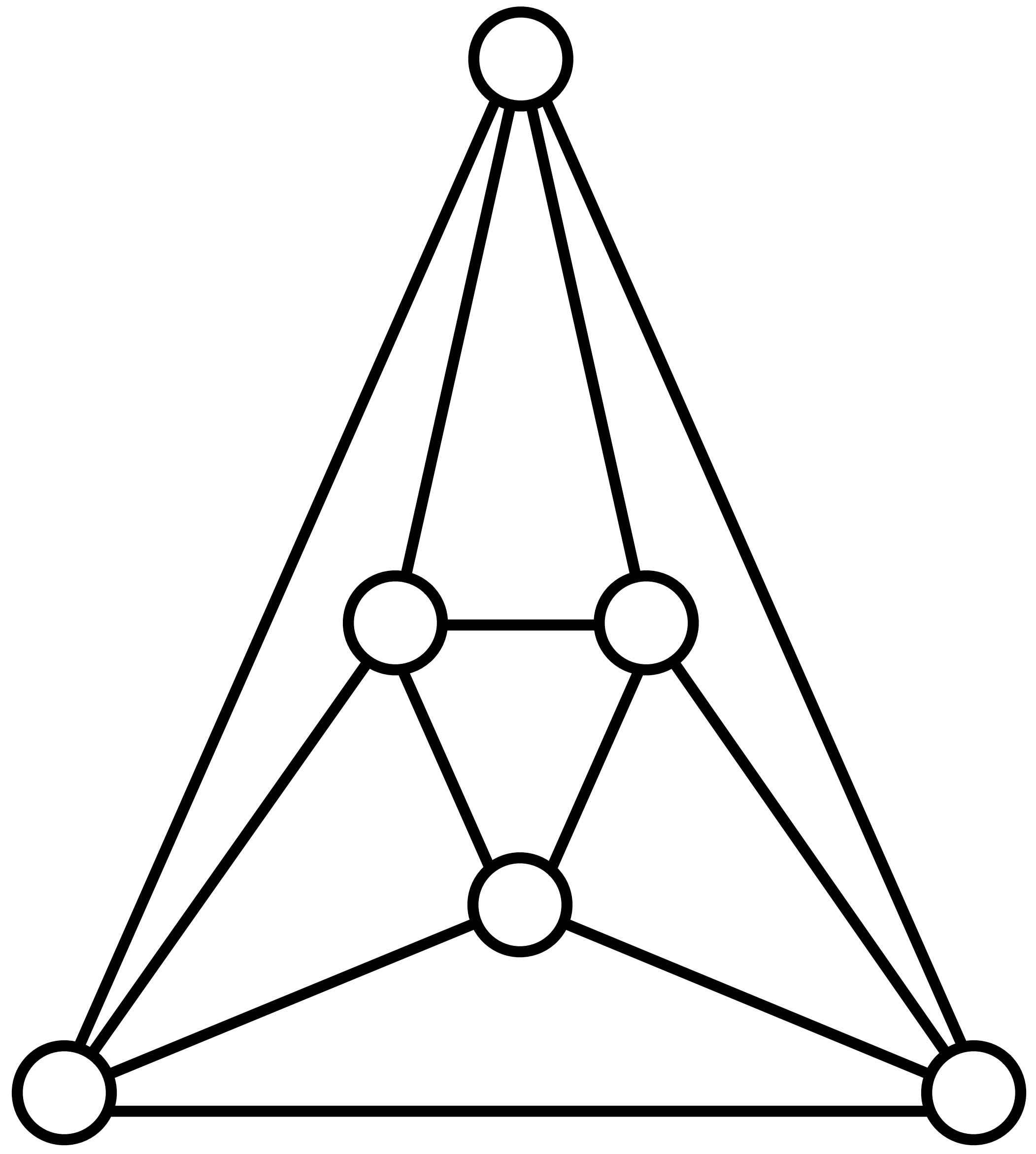
trees



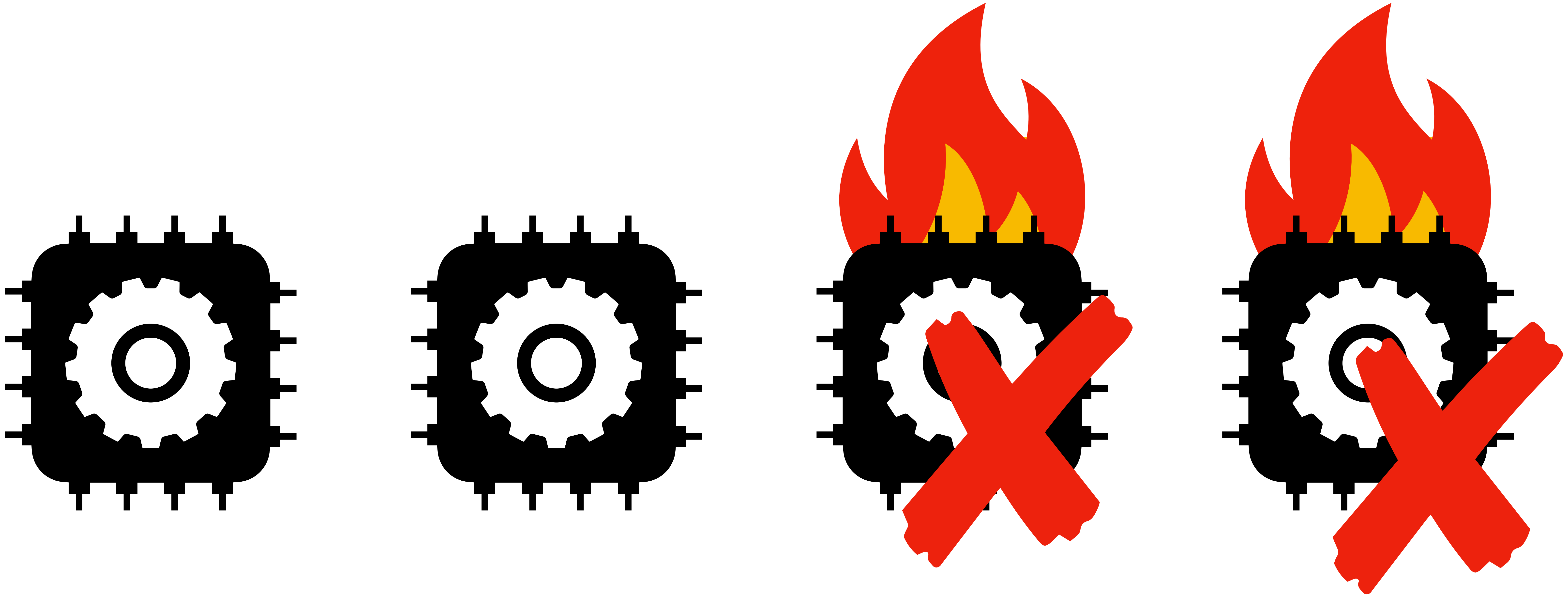
cycles



bridged graphs



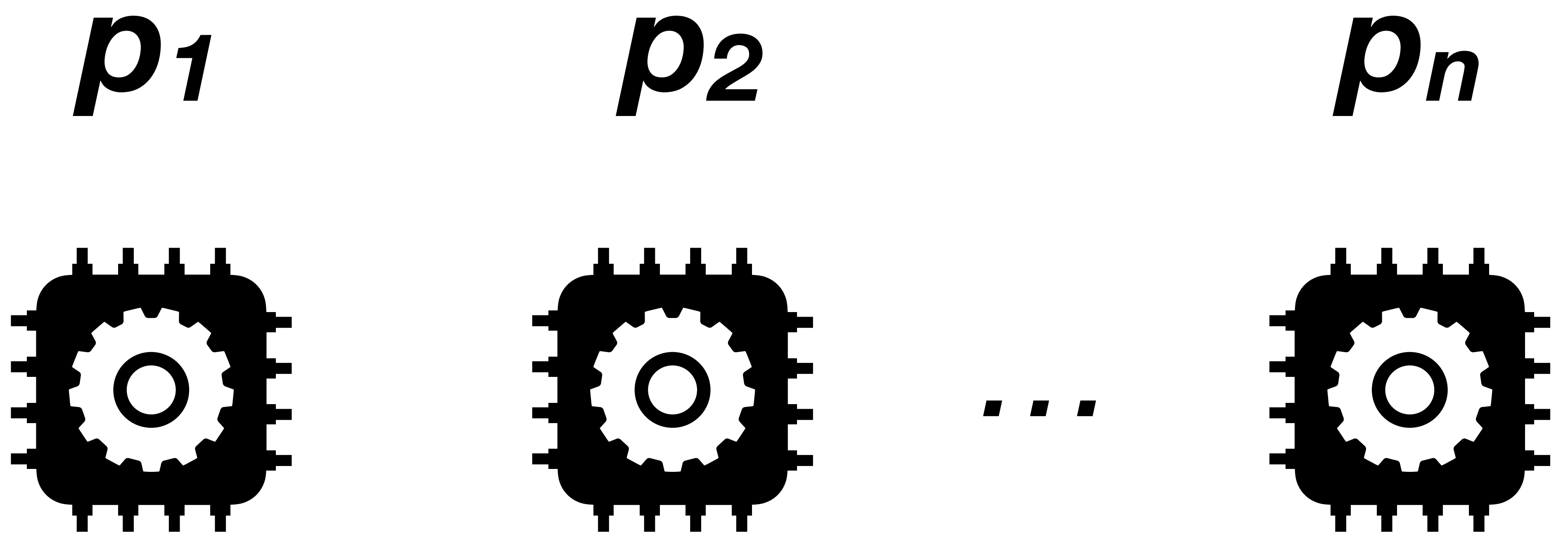
triangulated spheres



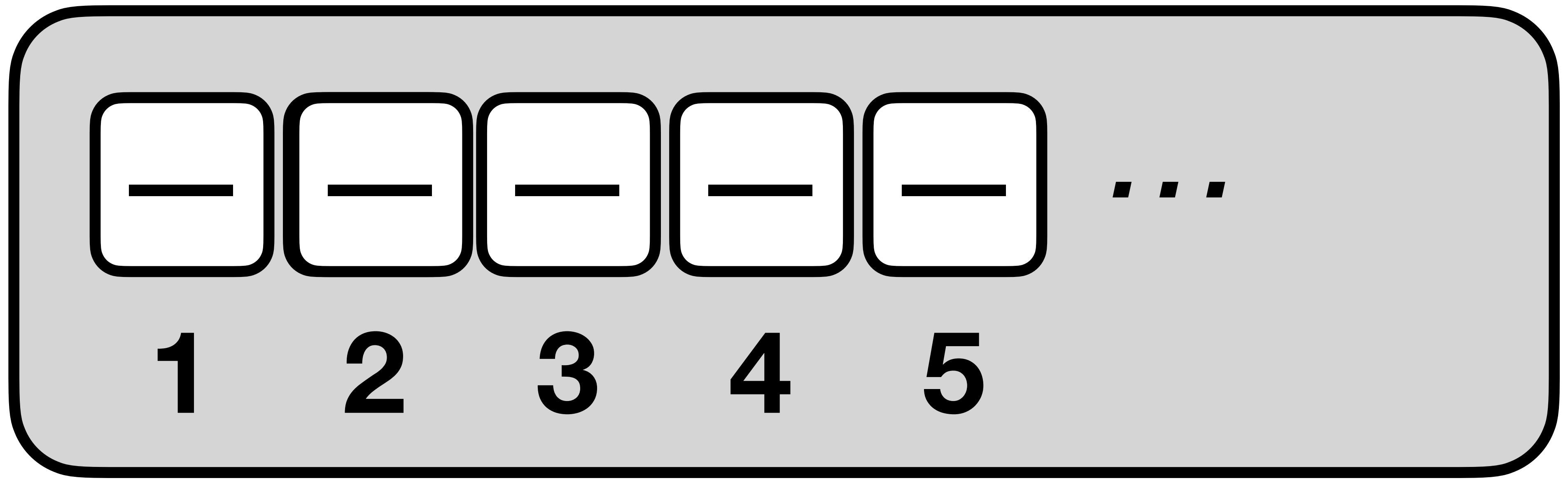
***k*-resilient:** despite at most *k* processes **crashing**,
correct processes terminate with correct outputs

**wait-free:** ($n$-1)-resilient

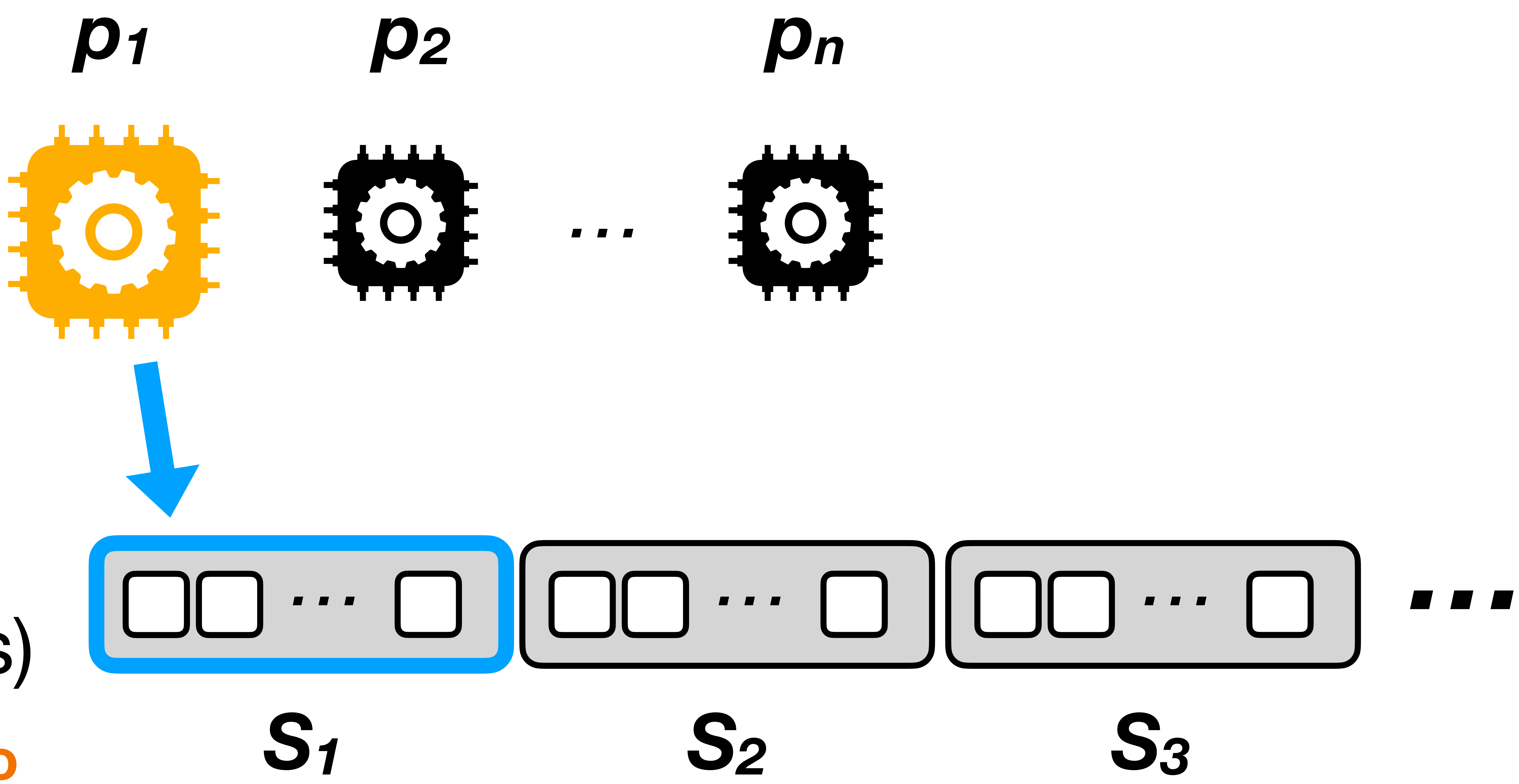
# **The model:**
asynchronous shared memory




**shared memory**
(registers)



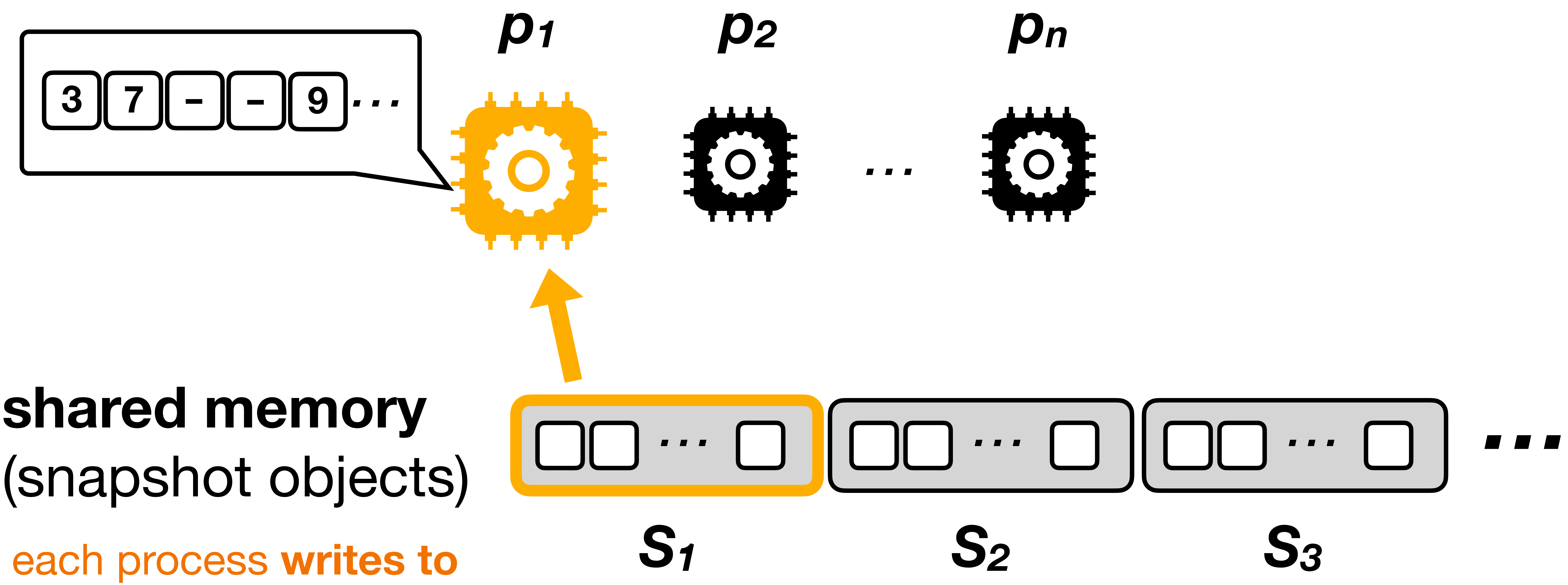
# The model:
## iterated snapshot model




**shared memory**
(snapshot objects)

each process **writes to** and **scans** each $S_i$ at most once

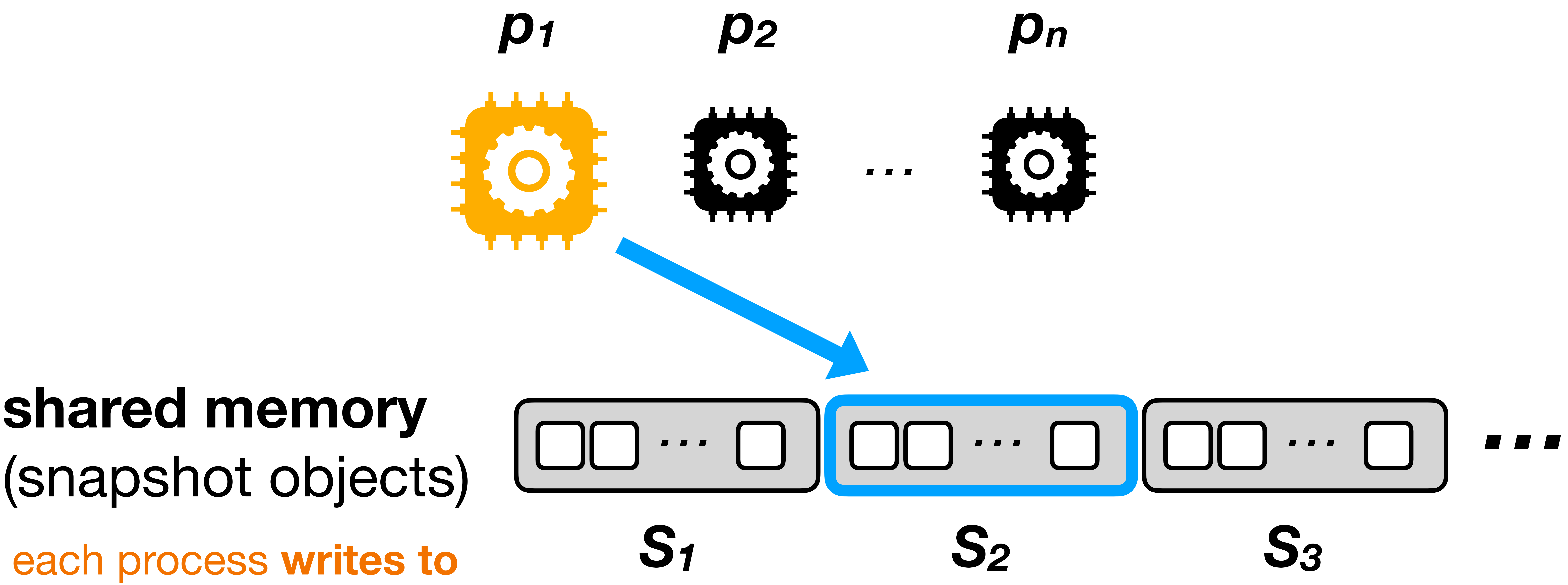
# **The model:**
iterated snapshot model




**shared memory**
(snapshot objects)

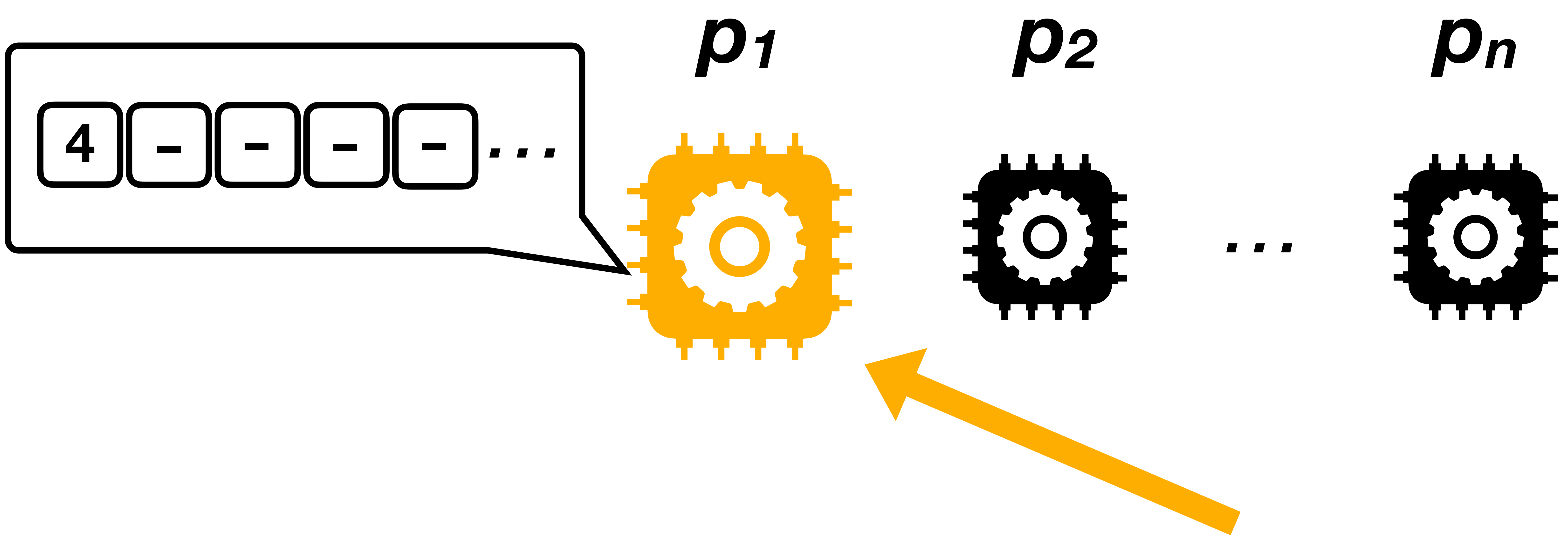
each process **writes to** and **scans** each $S_i$ at most once

# The model:
iterated snapshot model

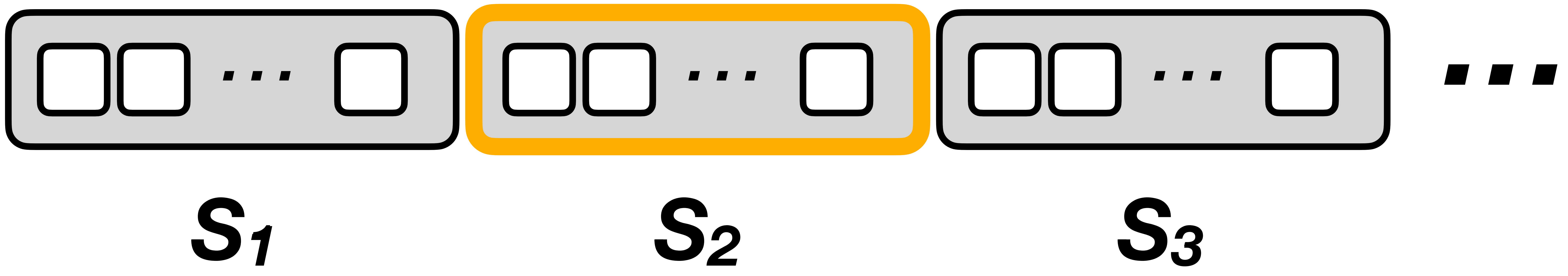



each process **writes to**
and **scans** each $S_i$ at most once

# The model:
iterated snapshot model

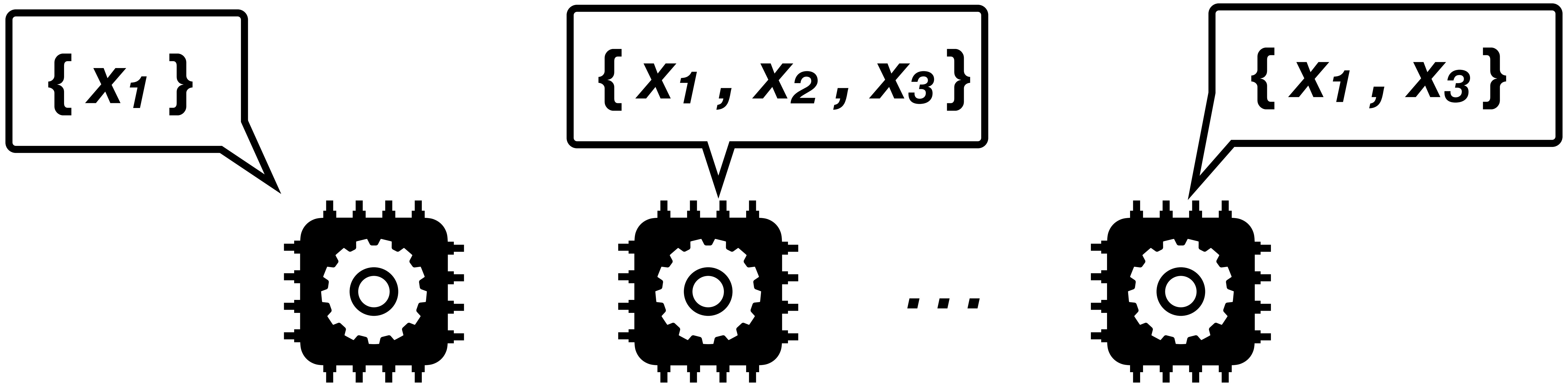


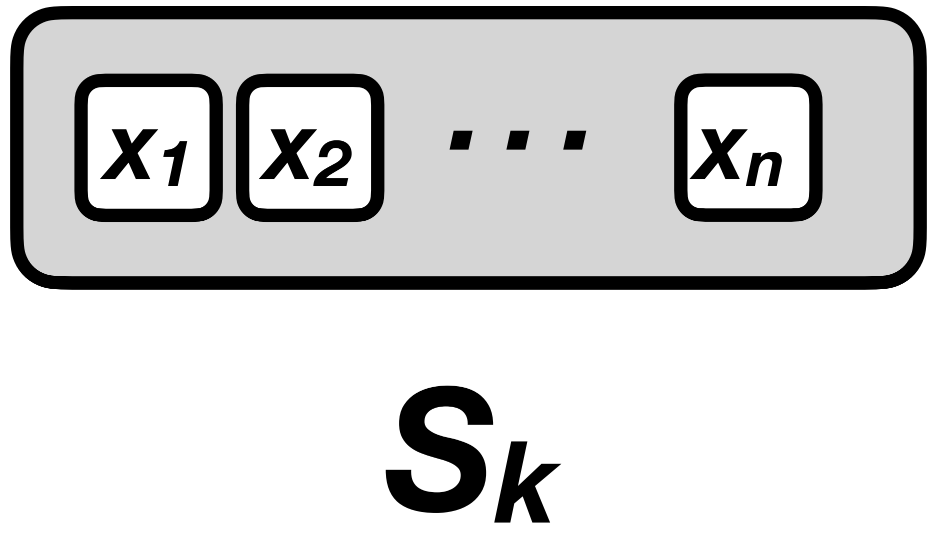

**shared memory**
(snapshot objects)




each process **writes to**
and **scans** each $S_i$ at most once

# **Scanning snapshots:**
# the containment property



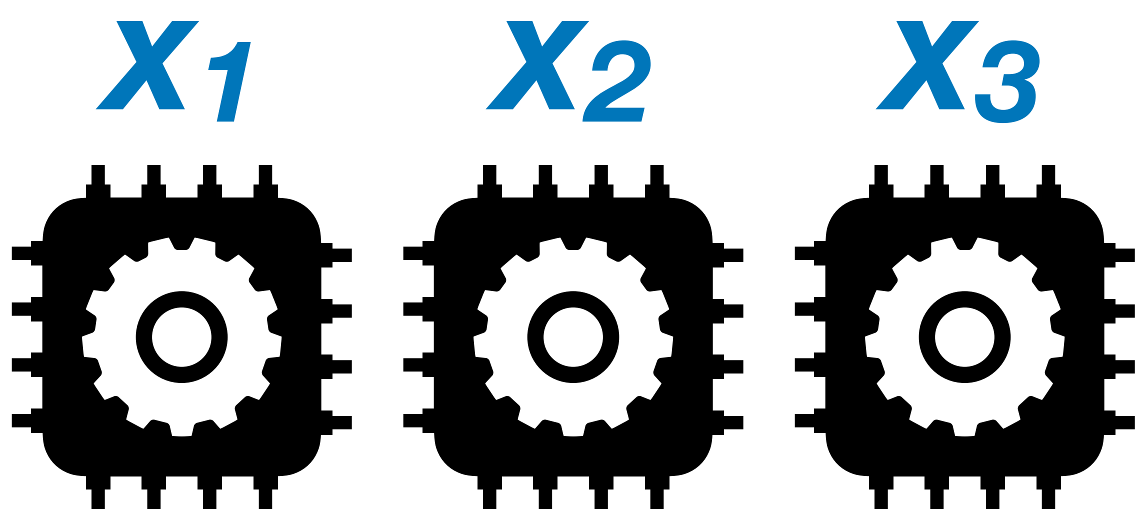



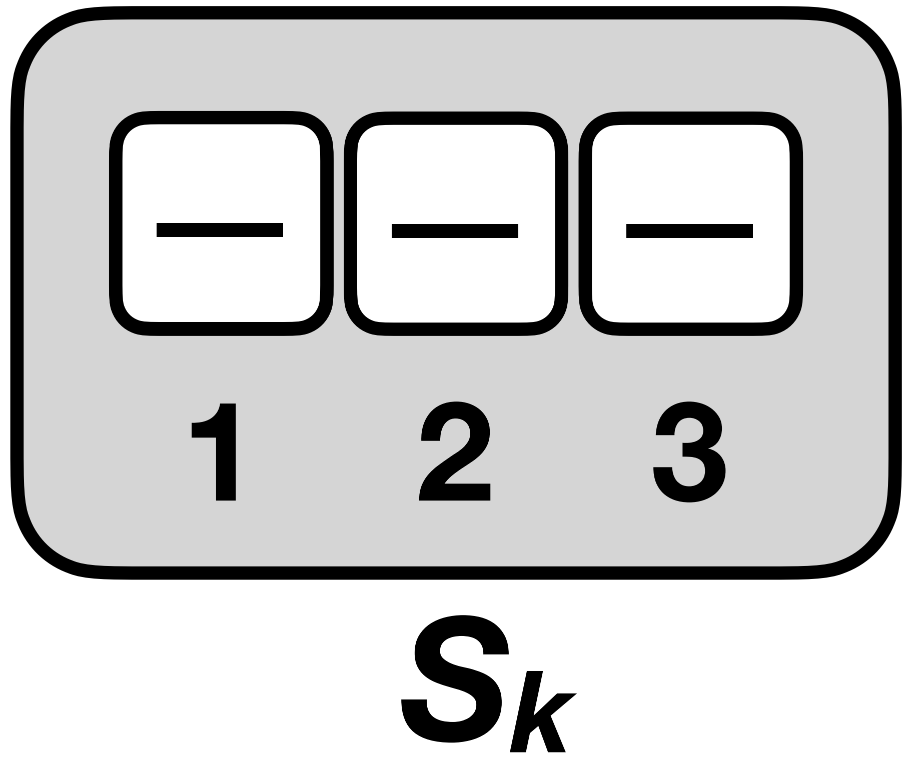

Containment property:

$$\{ x_1 \} \subseteq \{ x_1, x_3 \} \subseteq \{ x_1, x_2, x_3 \}$$

# The general algorithmic approach

# The general shape of algorithms



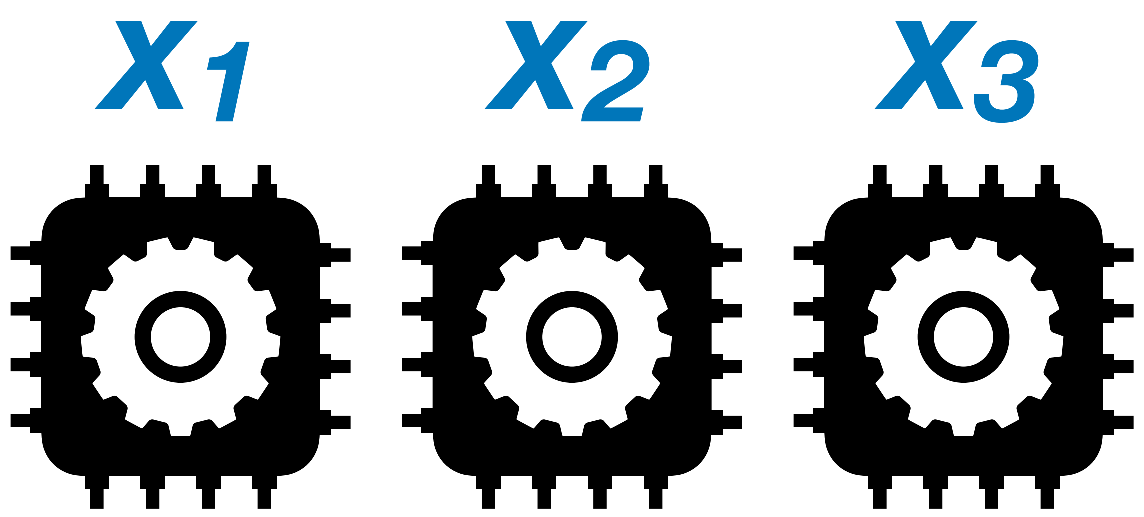




In iteration $k$ = 1, …, process $p_i$

1. writes $x_i$ to $S_k$
2. scans $S_k$ to obtain a set $V_i$
3. sets $x_i$ to $g( V_i )$

for suitable $g : 2^V \to V$

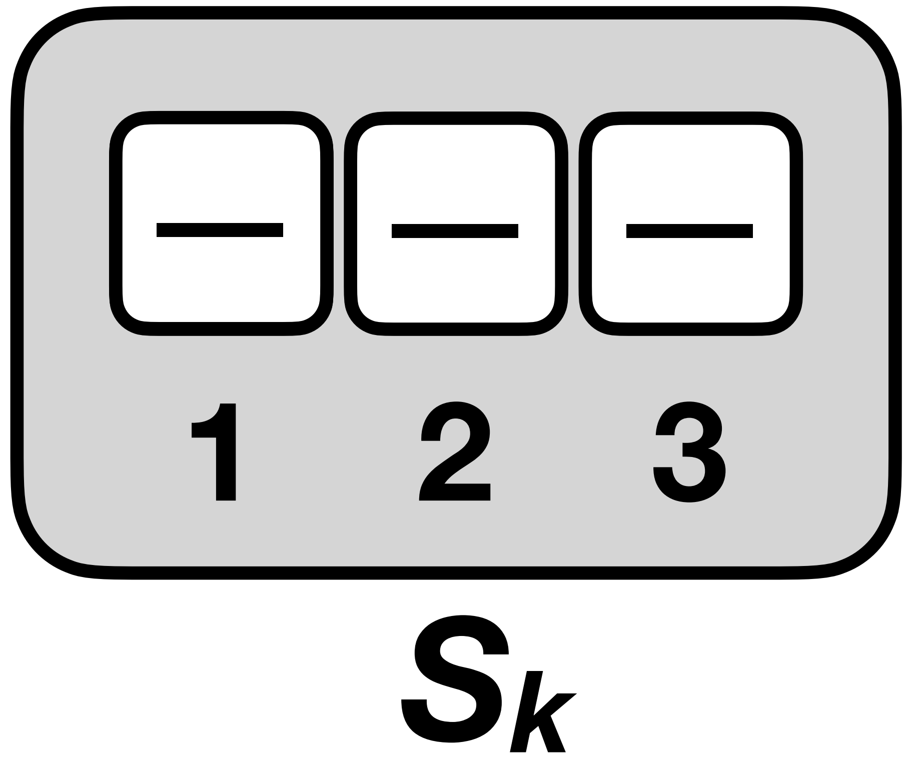
# The general shape of algorithms

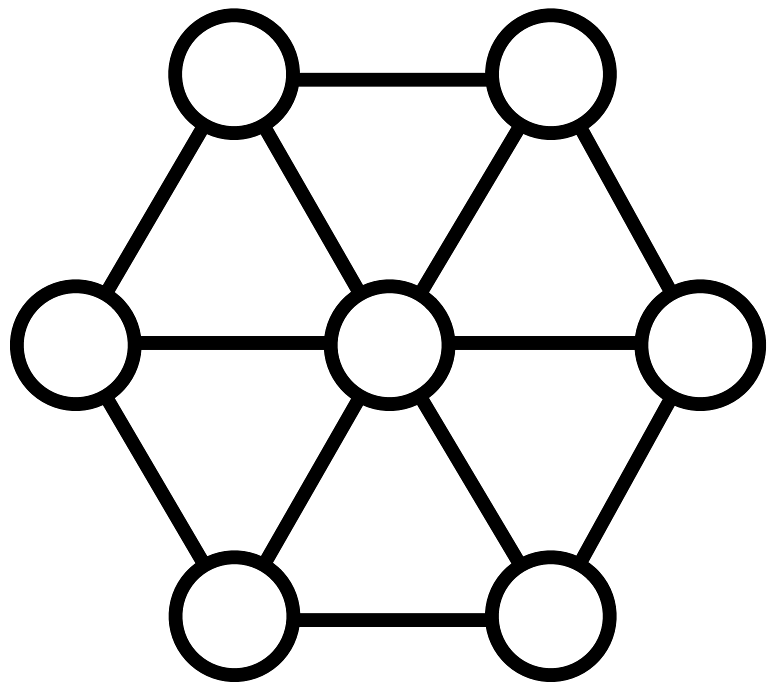



In iteration $k$ = 1, …, process $\boldsymbol{p_i}$

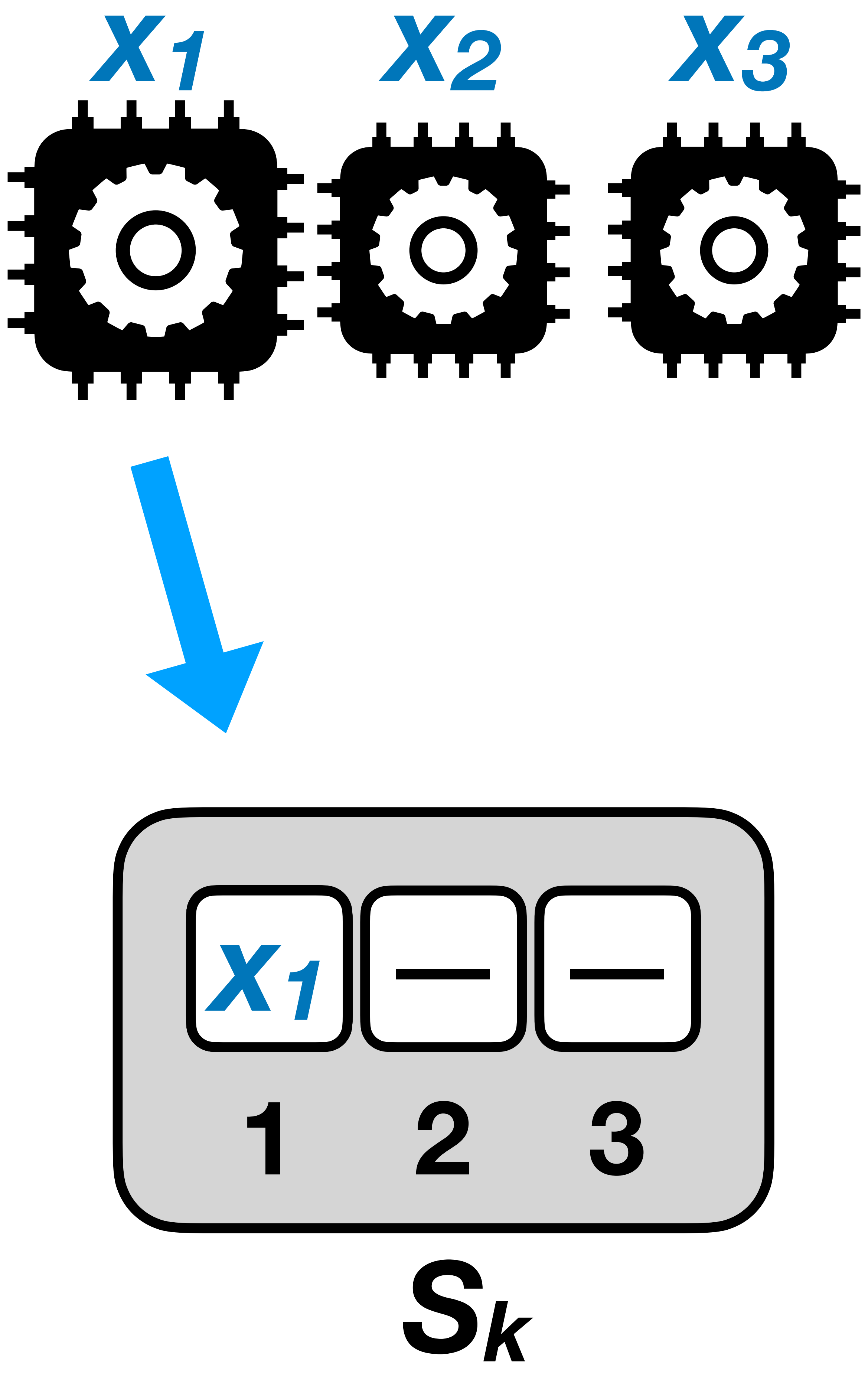
1. writes $\boldsymbol{x_i}$ to $\boldsymbol{S_k}$
2. scans $\boldsymbol{S_k}$ to obtain a set $\boldsymbol{V_i}$
3. sets $\boldsymbol{x_i}$ to $\boldsymbol{g(V_i)}$

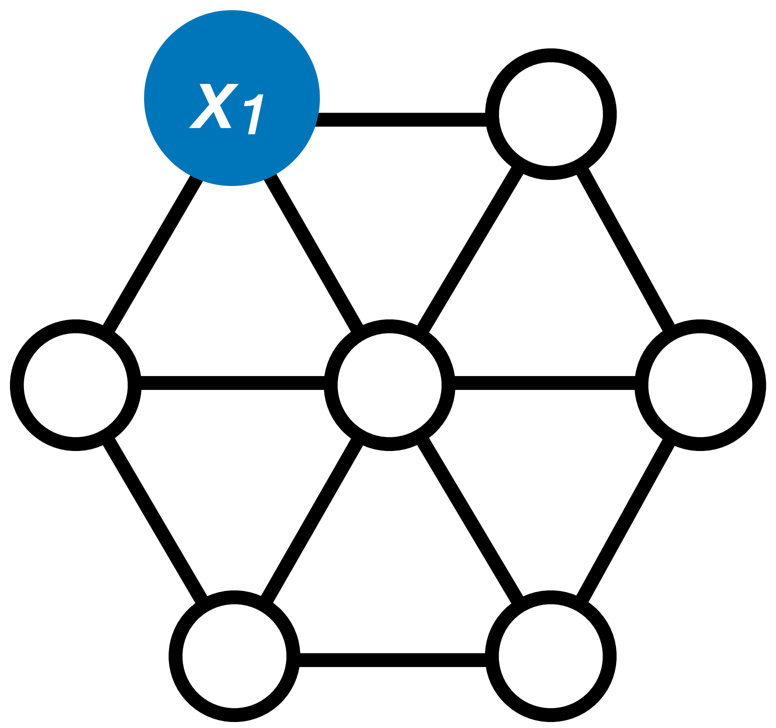
for suitable $\boldsymbol{g : 2^V \to V}$



# The general shape of algorithms



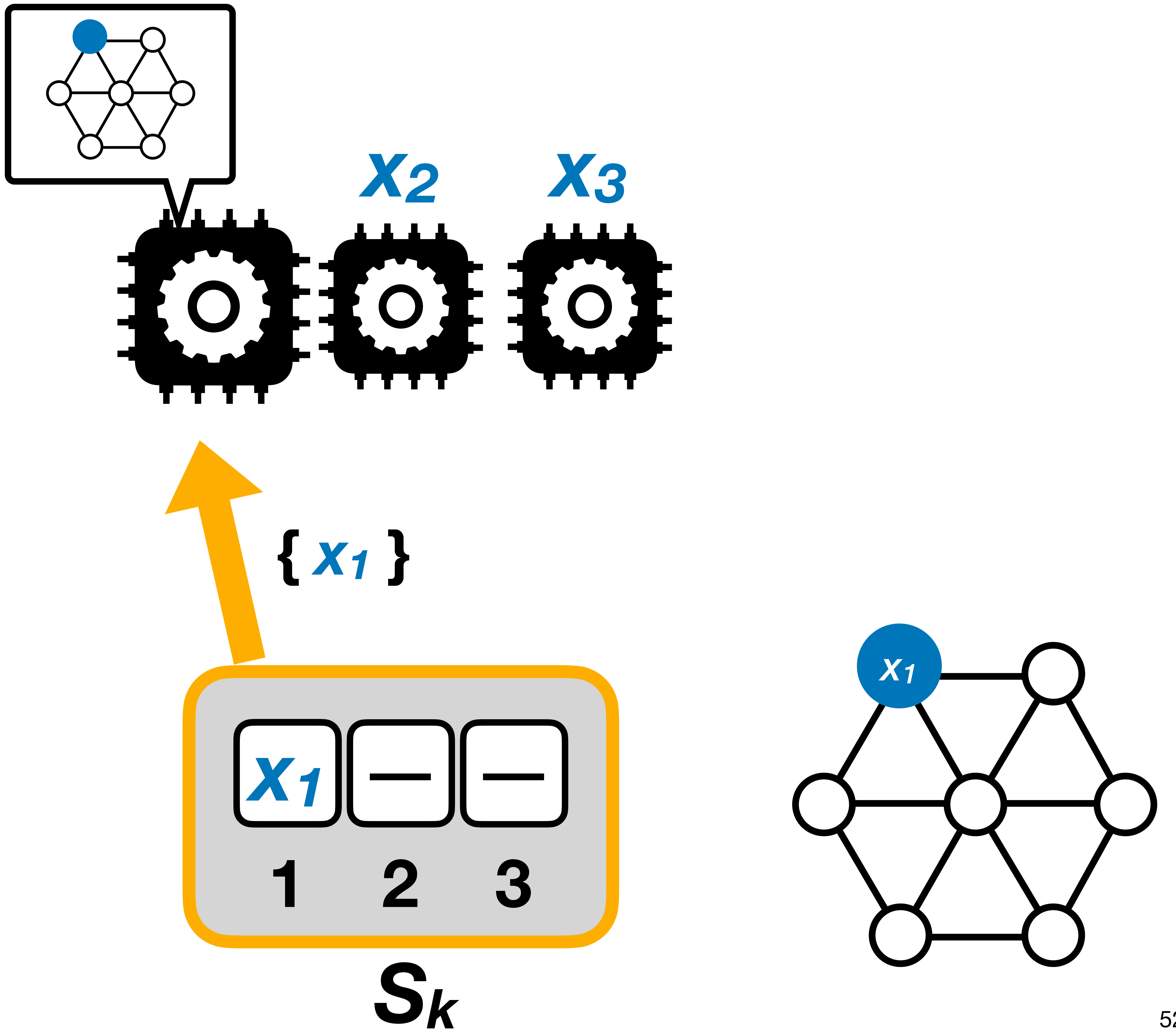




In iteration $k = 1, \ldots,$ process $\boldsymbol{p_i}$

1. writes $\boldsymbol{x_i}$ to $\boldsymbol{S_k}$
2. scans $\boldsymbol{S_k}$ to obtain a set $\boldsymbol{V_i}$
3. sets $\boldsymbol{x_i}$ to $\boldsymbol{g(\, V_i\, )}$

for suitable $\boldsymbol{g : 2^V \to V}$

# The general shape of algorithms

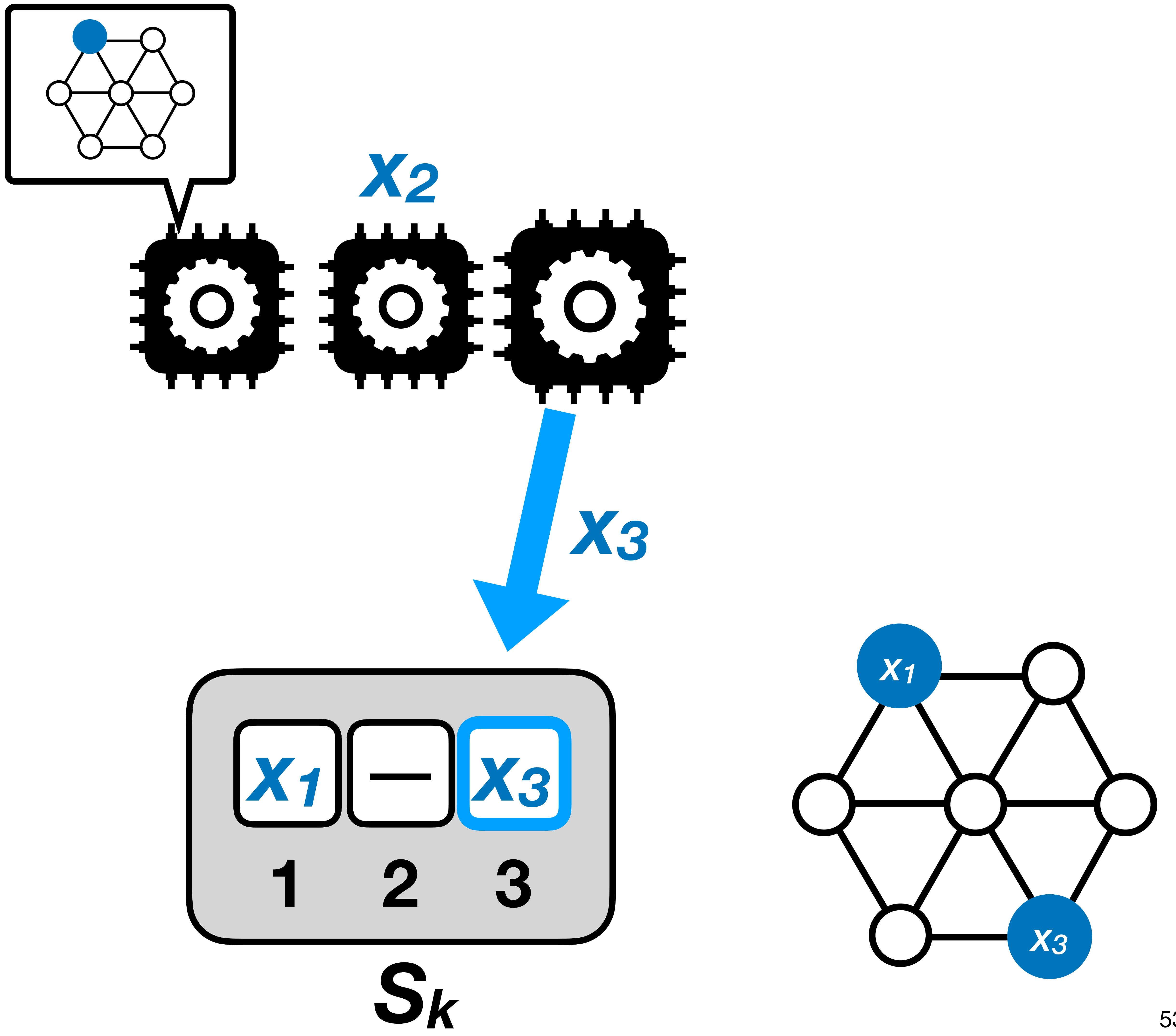



In iteration $k = 1, \ldots,$ process $p_i$

1. writes $x_i$ to $S_k$
2. scans $S_k$ to obtain a set $V_i$
3. sets $x_i$ to $g( V_i )$

for suitable $g : 2^V \to V$

# The general shape of algorithms

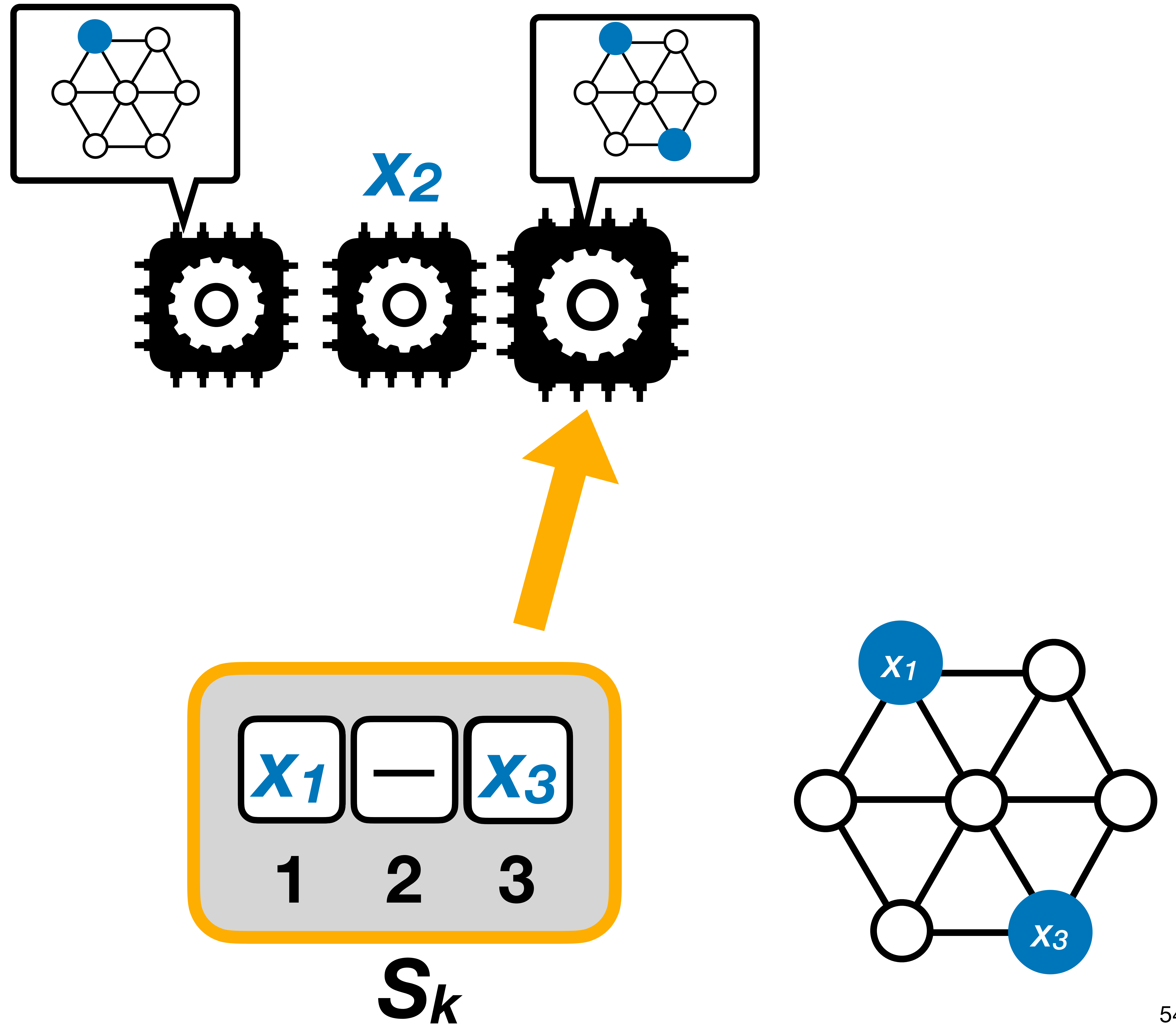



In iteration $k$ = 1, …, process $p_i$

1. writes $x_i$ to $S_k$
2. scans $S_k$ to obtain a set $V_i$
3. sets $x_i$ to $g( V_i )$

for suitable $g : 2^V \to V$

# The general shape of algorithms

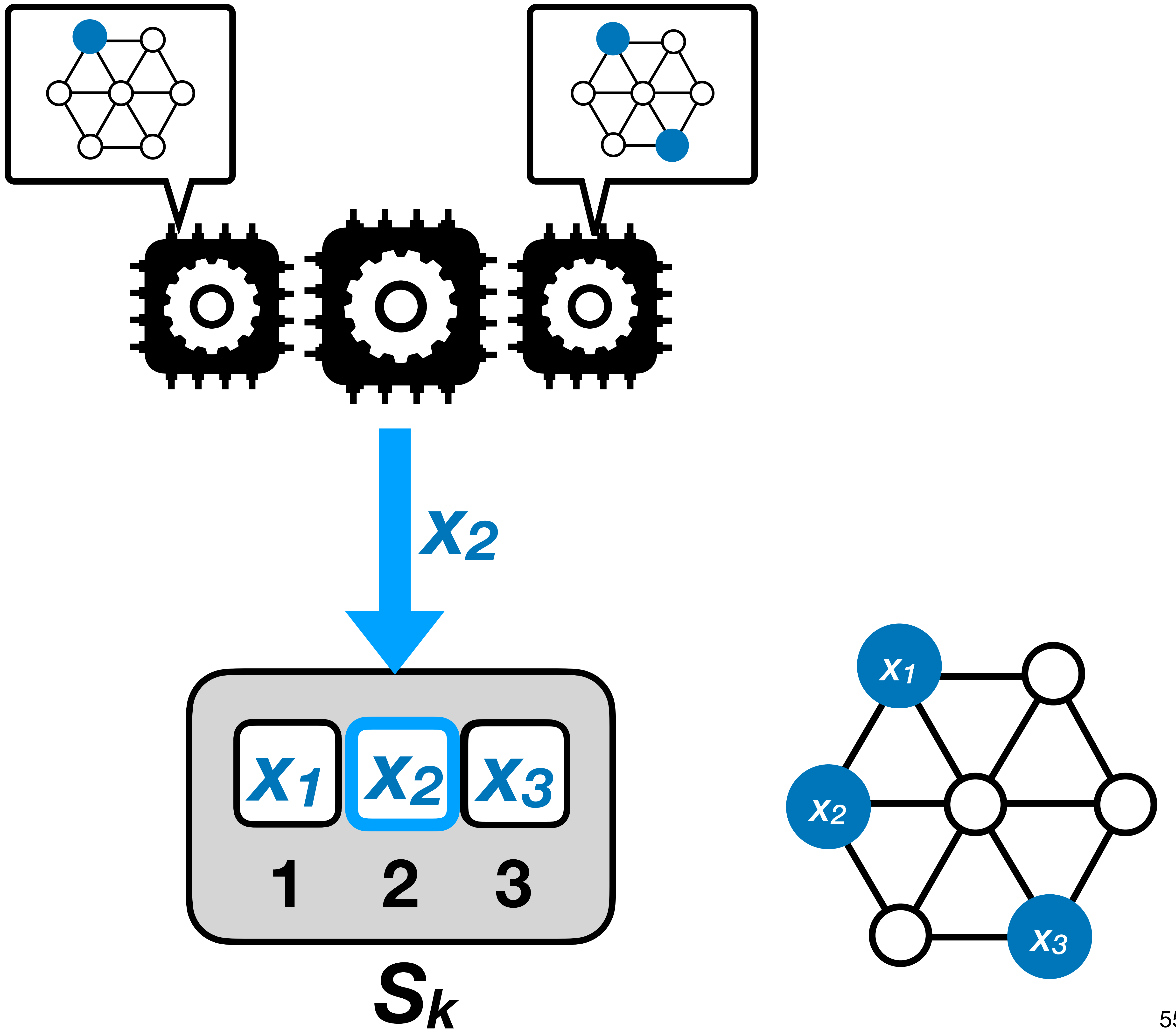



In iteration $k = 1, \ldots,$ process $\boldsymbol{p_i}$

1. writes $\boldsymbol{x_i}$ to $\boldsymbol{S_k}$
2. scans $\boldsymbol{S_k}$ to obtain a set $\boldsymbol{V_i}$
3. sets $\boldsymbol{x_i}$ to $\boldsymbol{g(\ V_i\ )}$

for suitable $\boldsymbol{g : 2^V \rightarrow V}$

# The general shape of algorithms

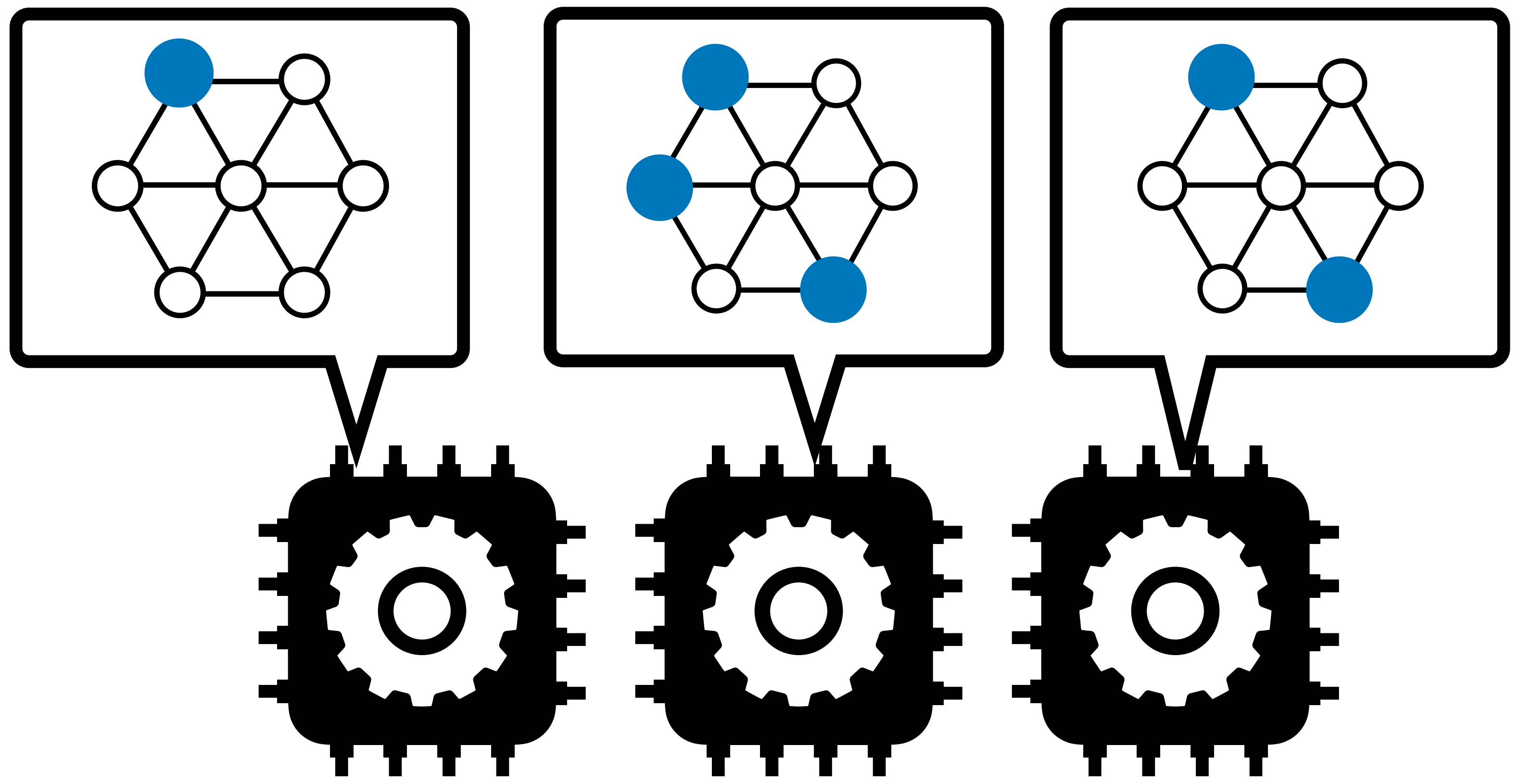


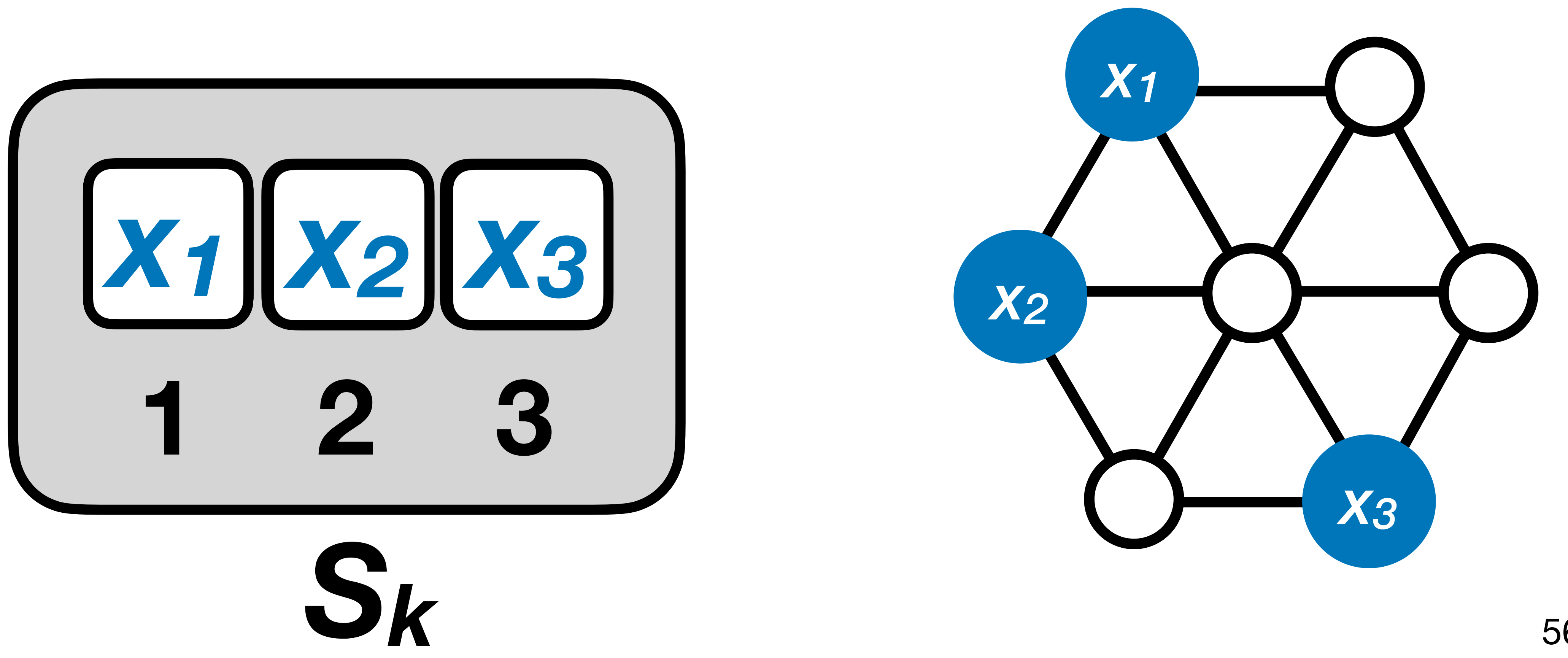

In iteration $k$ = 1, …, process $\boldsymbol{p_i}$

1. writes $\boldsymbol{x_i}$ to $\boldsymbol{S_k}$
2. scans $\boldsymbol{S_k}$ to obtain a set $\boldsymbol{V_i}$
3. sets $\boldsymbol{x_i}$ to $\boldsymbol{g(\ V_i\ )}$

for suitable $\boldsymbol{g : 2^V \to V}$

# The general shape of algorithms



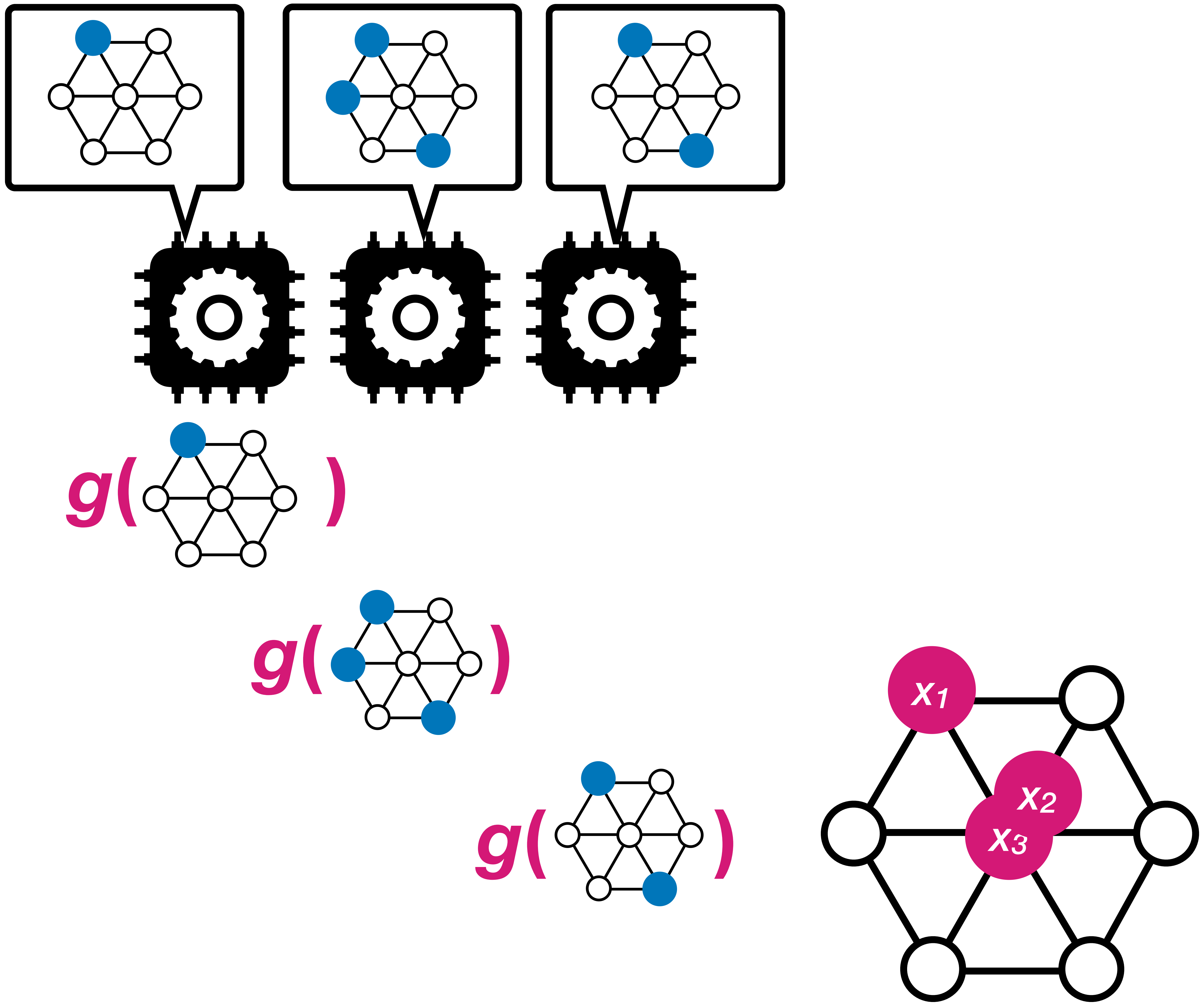



In iteration $k$ = 1, …, process $\boldsymbol{p_i}$

1. writes $\boldsymbol{x_i}$ to $\boldsymbol{S_k}$
2. scans $\boldsymbol{S_k}$ to obtain a set $\boldsymbol{V_i}$
3. sets $\boldsymbol{x_i}$ to $\boldsymbol{g(\ V_i\ )}$

for suitable $\boldsymbol{g : 2^V \rightarrow V}$

# The general shape of algorithms

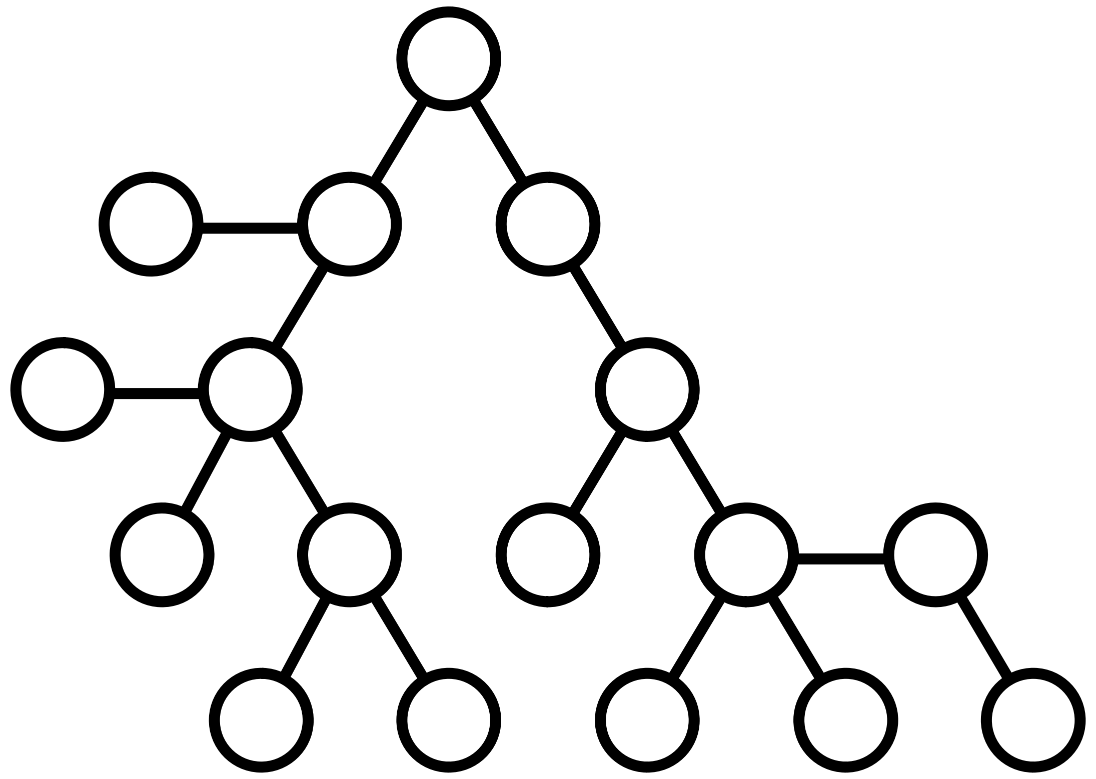



In iteration $k$ = 1, …, process $p_i$

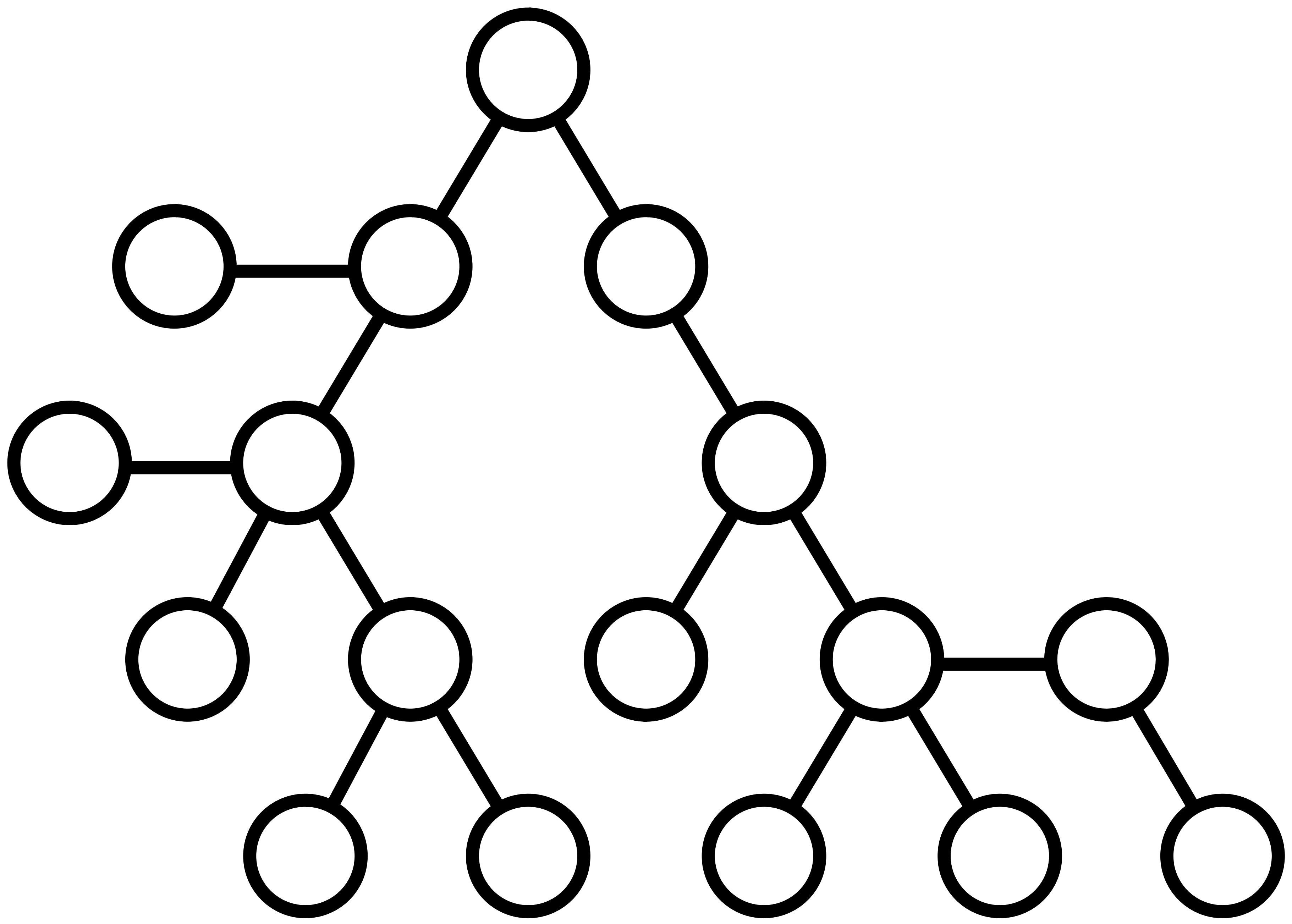
1. writes $x_i$ to $S_k$
2. scans $S_k$ to obtain a set $V_i$
3. sets $x_i$ to $g(\ V_i\ )$

for suitable $g : 2^V \to V$

# A wait-free algorithm for trees

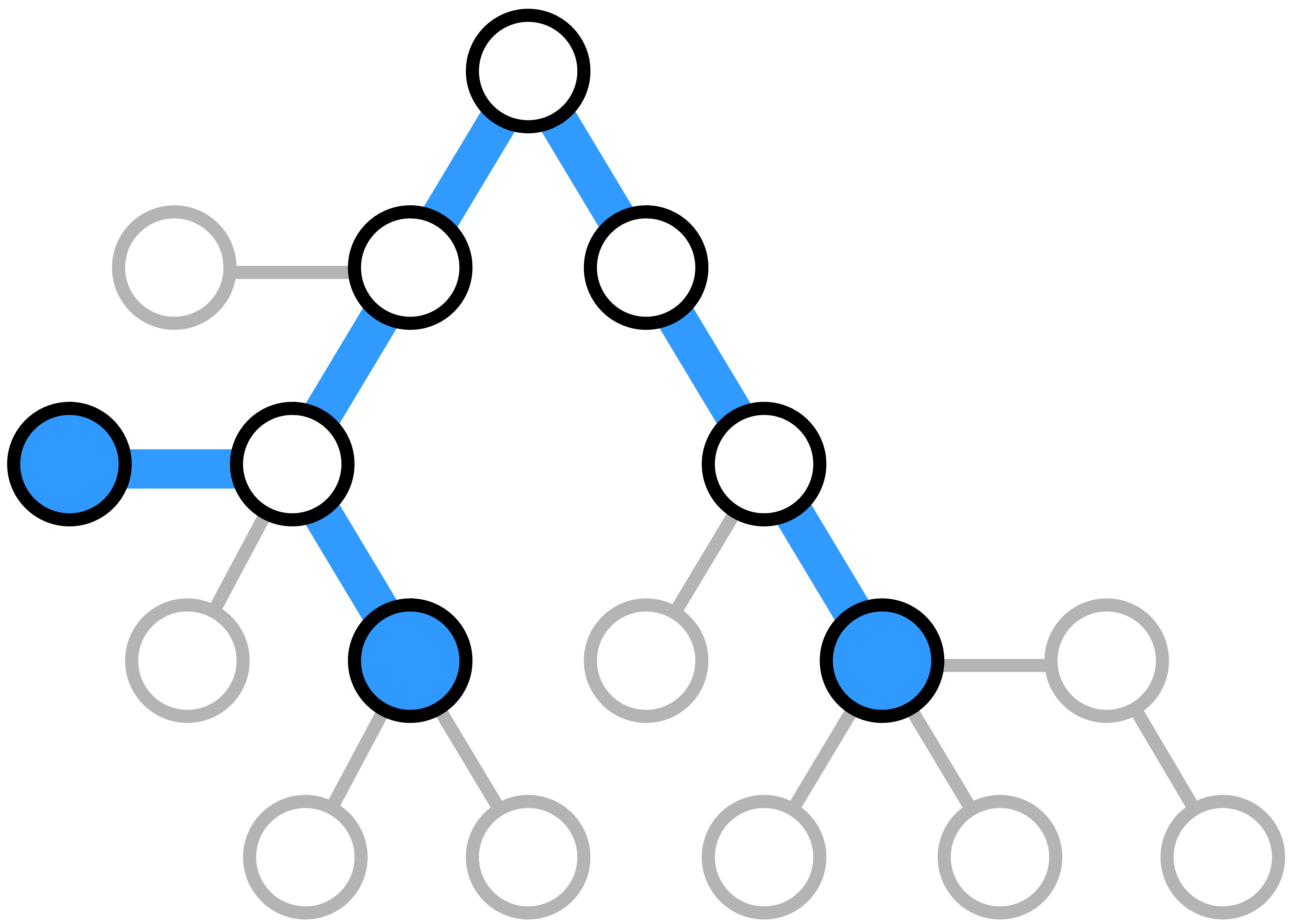


# Algorithm for trees



# Algorithm for trees

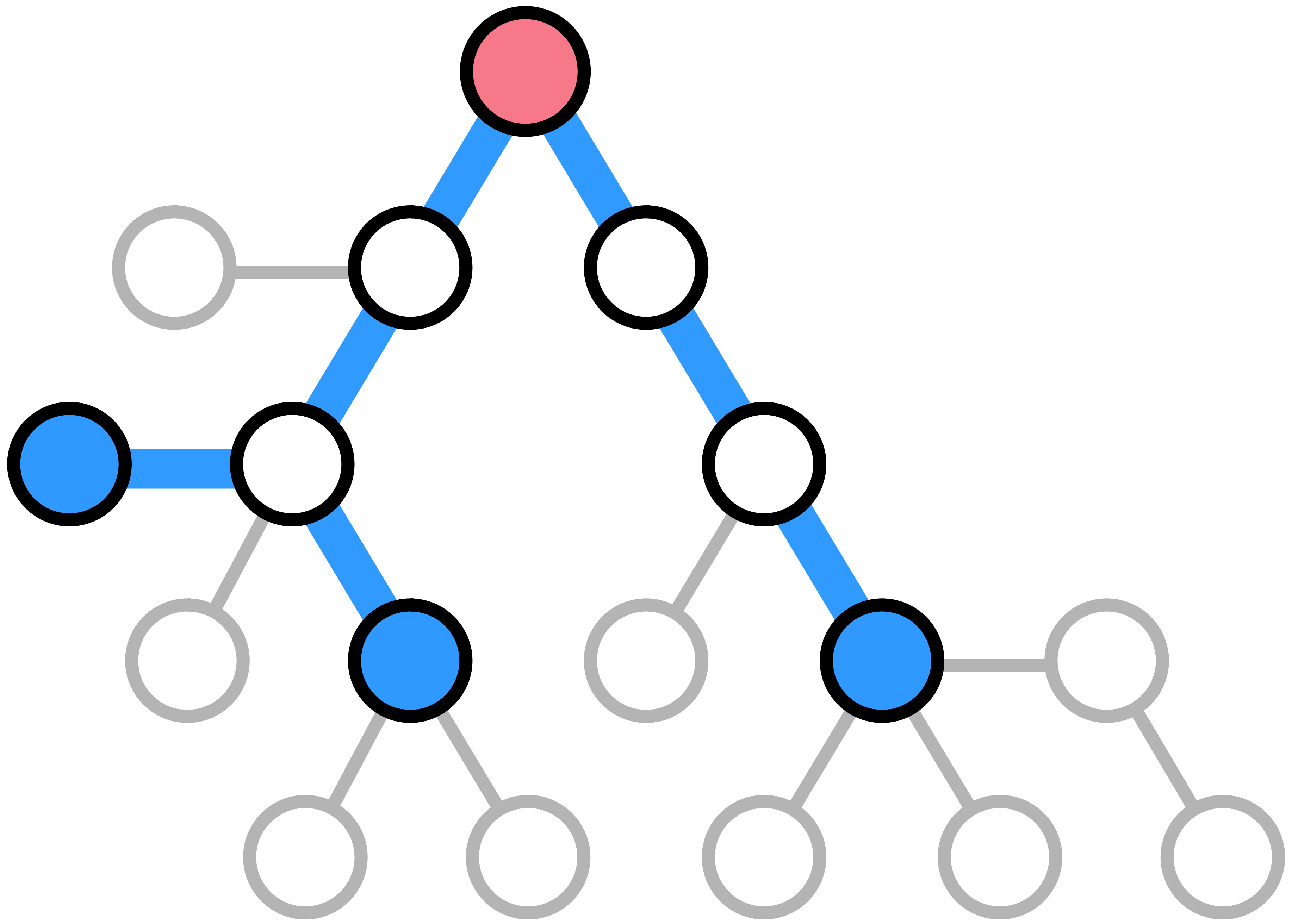
The ***convex hull*** **<** ***U*** **>** of ***U*** consists of all vertices on shortest paths between vertices of ***U*****.**



# Algorithm for trees

The ***convex hull*** **<** ***U*** **>** of ***U*** consists of all vertices on shortest paths between vertices of ***U.***

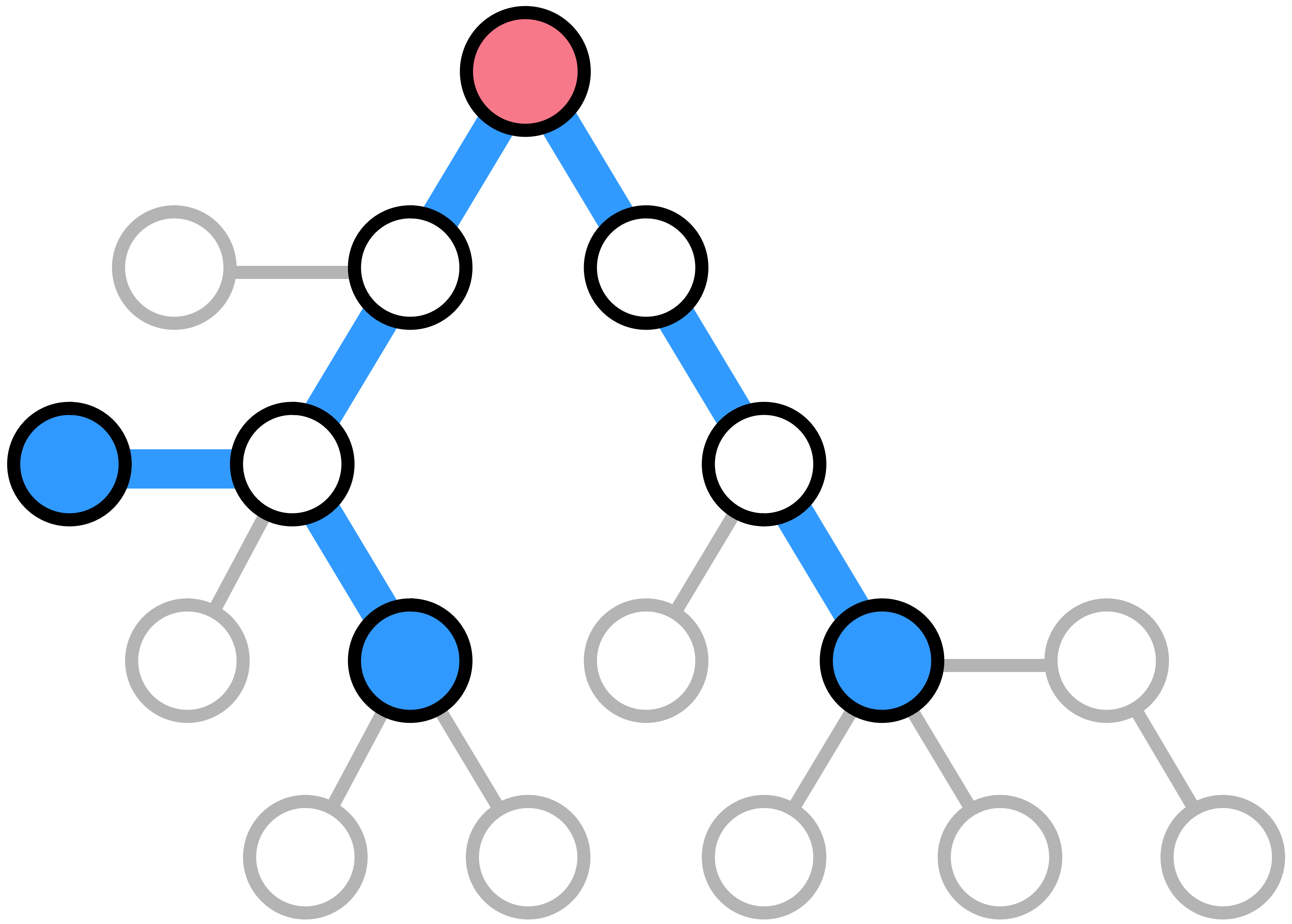
The ***center*** of **<** ***U*** **>** is the *set* of vertices that minimise the maximum distance to any other node in **<** ***U*** **>.**



# Algorithm for trees

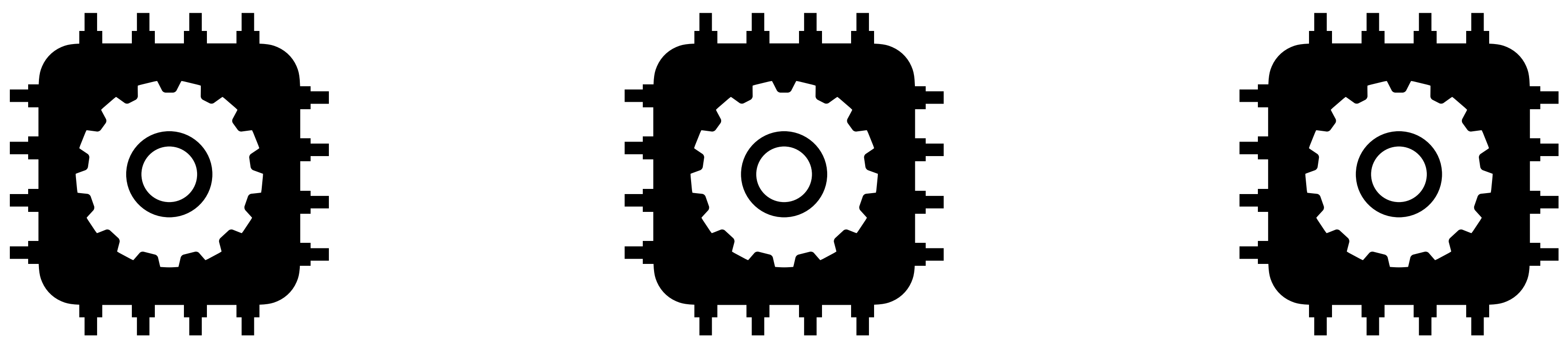
The ***convex hull* < *U* >** of ***U*** consists of all vertices on shortest paths between vertices of ***U*.**

The *center* of **< *U* >** is the *set* of vertices that minimise the maximum distance to any other node in **< *U* >.**

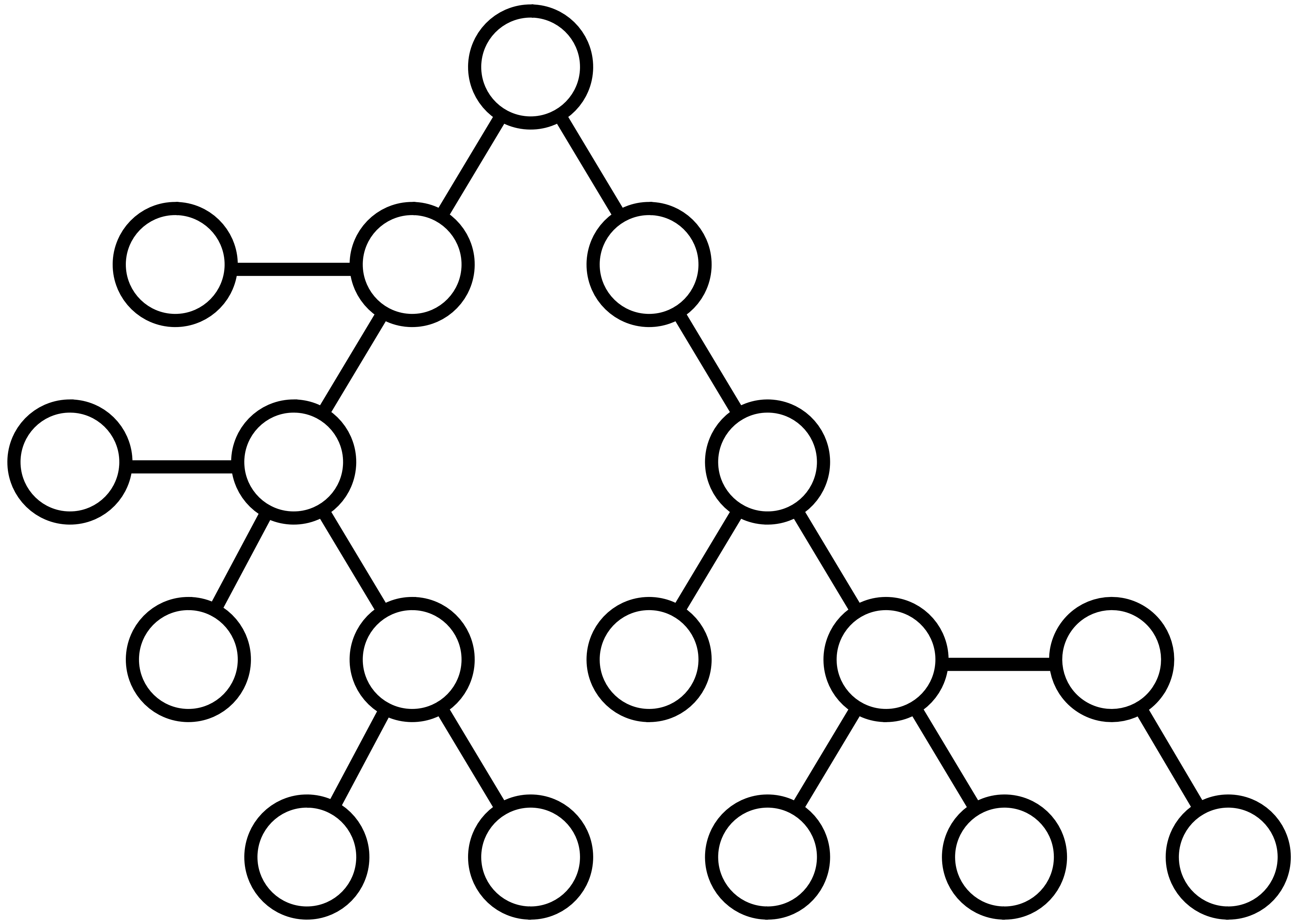


**Update rule:**
let ***g*( *U* )** to be a vertex in the *center* of **< *U* >**

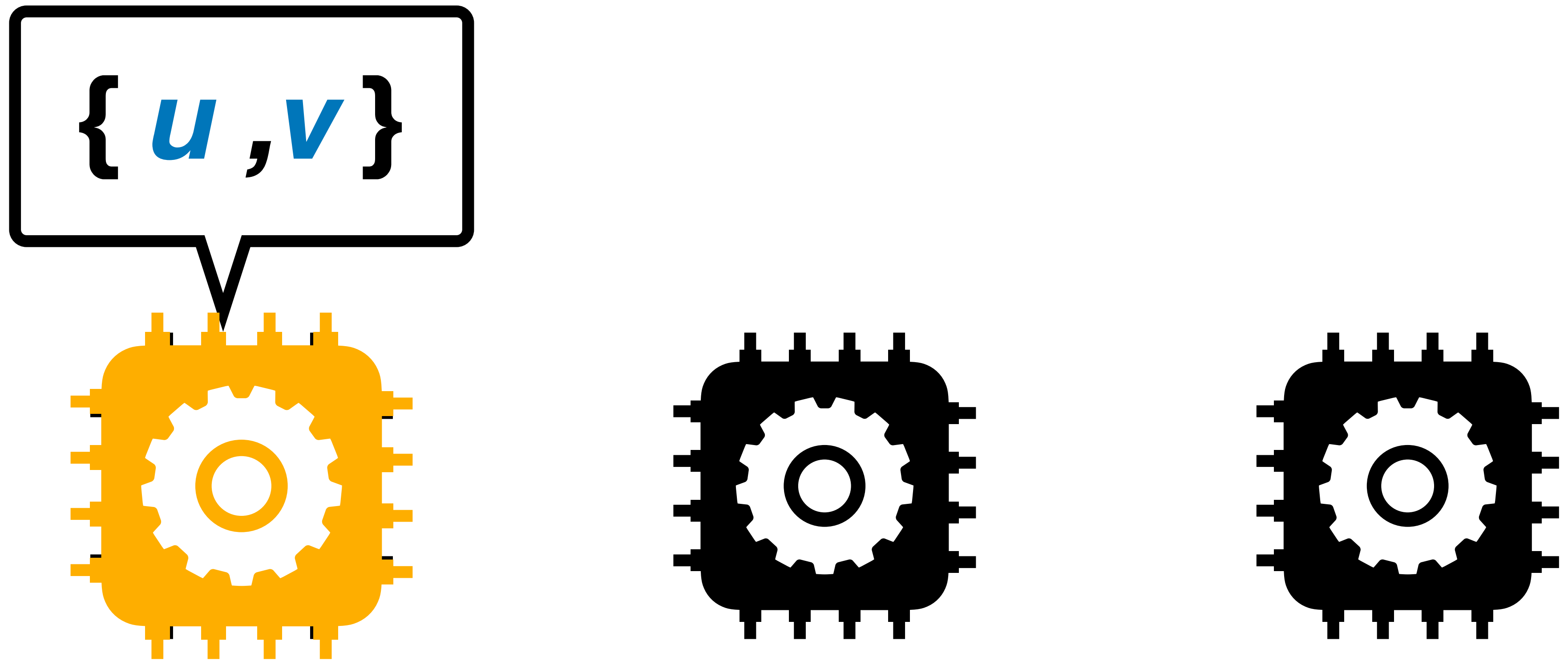
# Algorithm for trees



**Update rule:**
let ***g( U )*** to be a vertex in the ***center*** of **< *U* >**

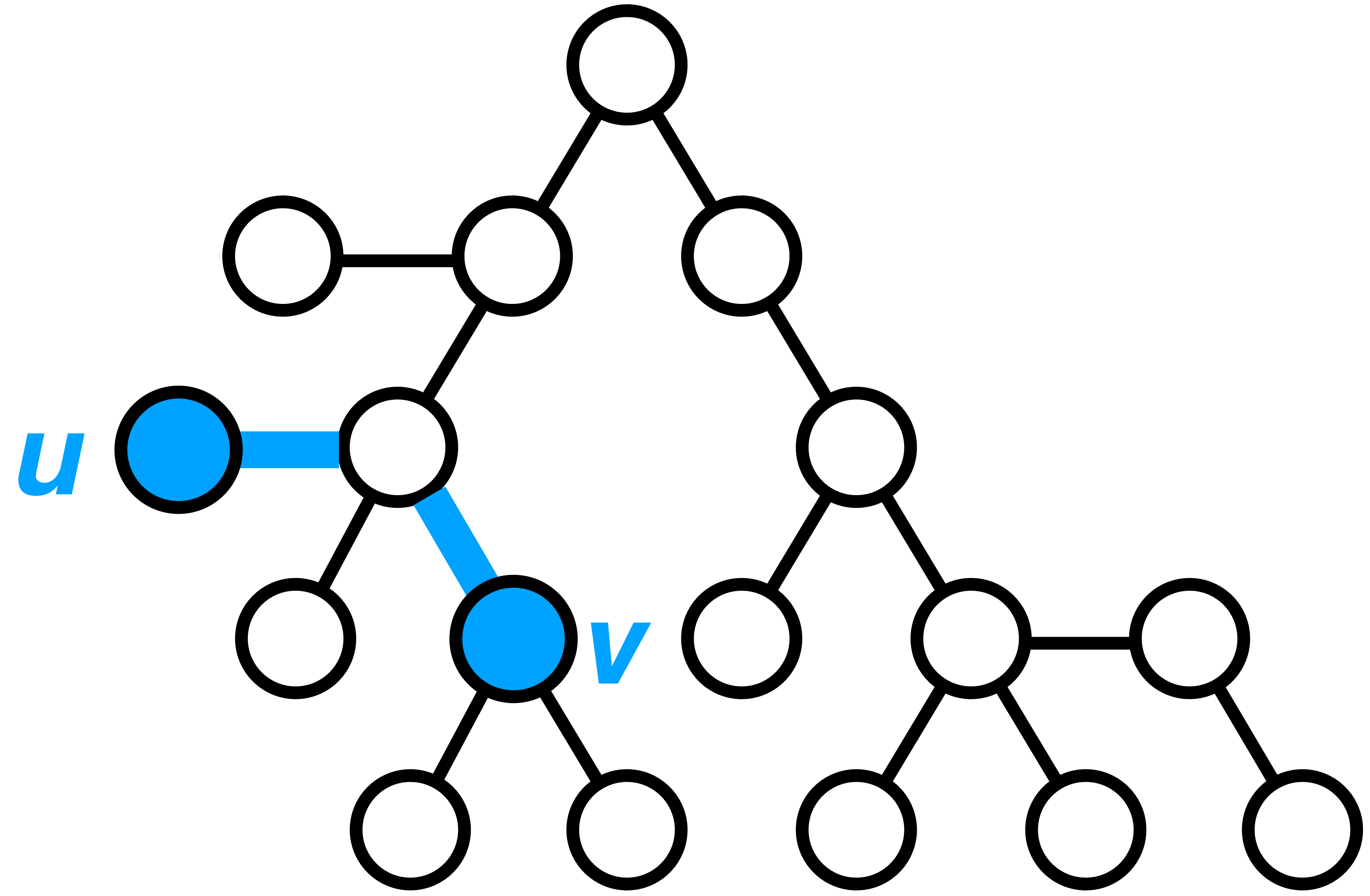


# Algorithm for trees



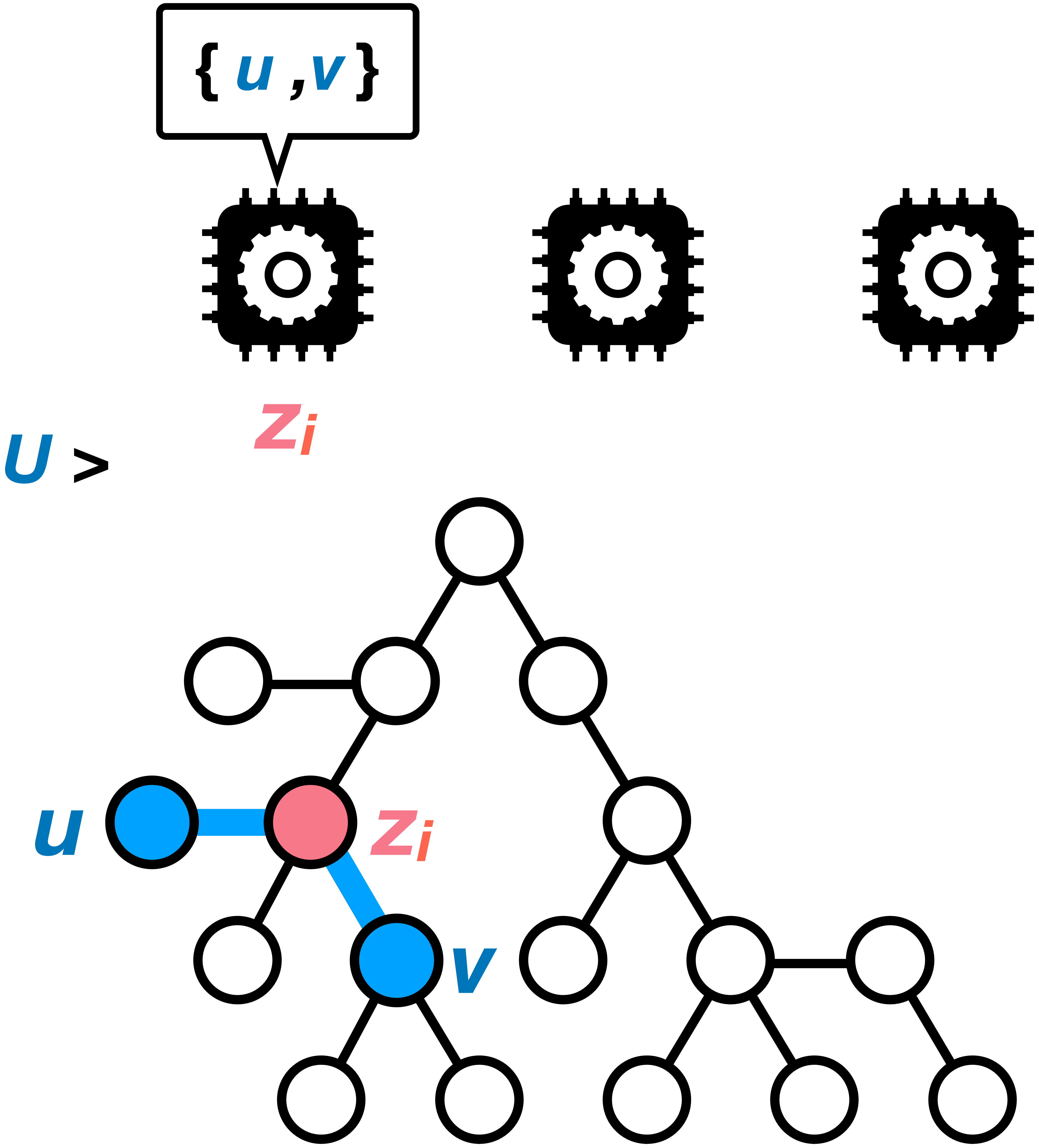

**Update rule:**

let *g( U )* to be a vertex in the *center* of $< U >$



# Algorithm for trees

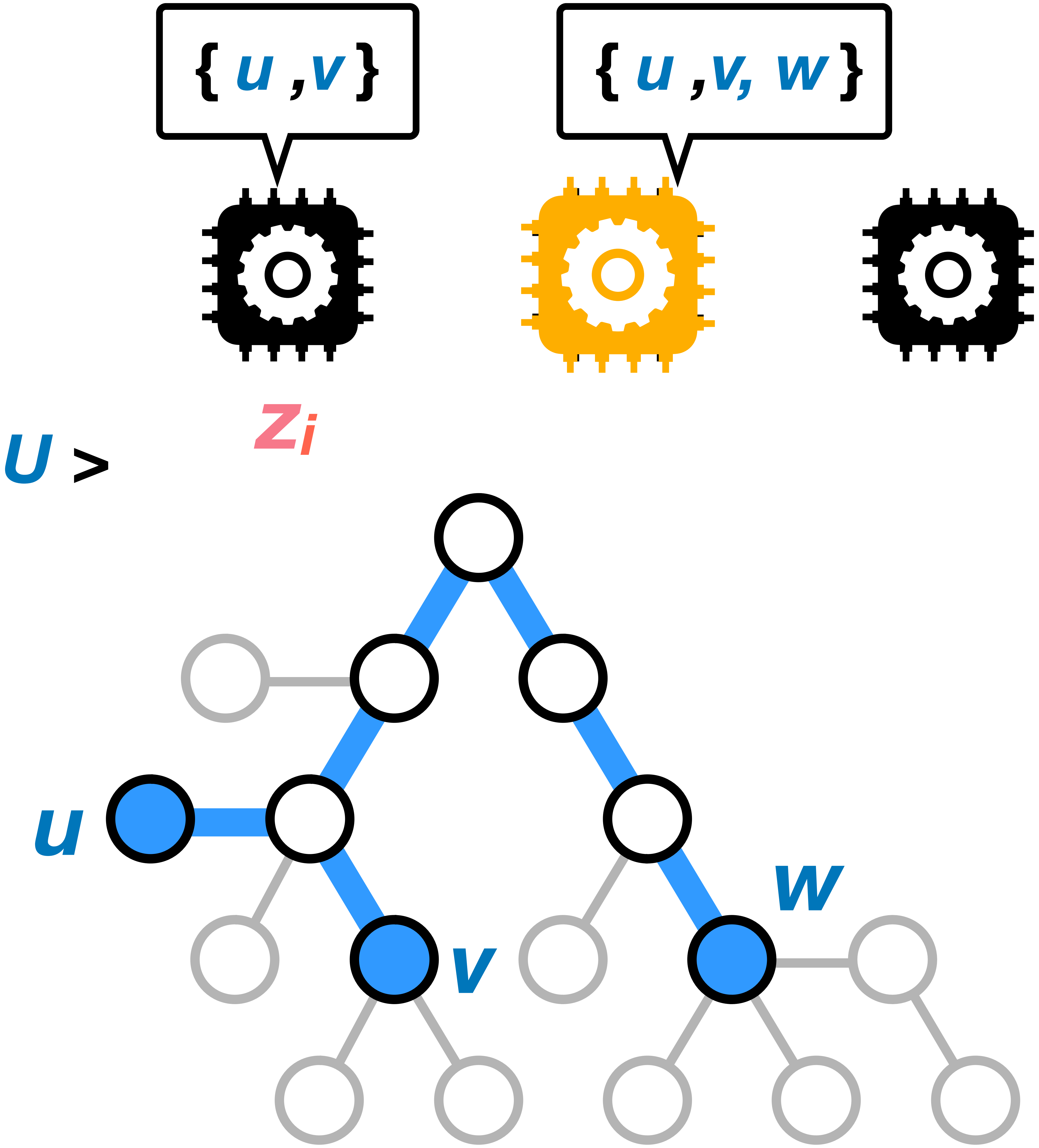
**Update rule:**
let $g(U)$ to be a vertex in the *center* of $< U >$



# Algorithm for trees

**Update rule:**
let $g(U)$ to be a vertex in the *center* of $< U >$

**Containment property:**
$V_{j(1)} \subseteq V_{j(2)} \subseteq \ldots \subseteq V_{j(n)}$



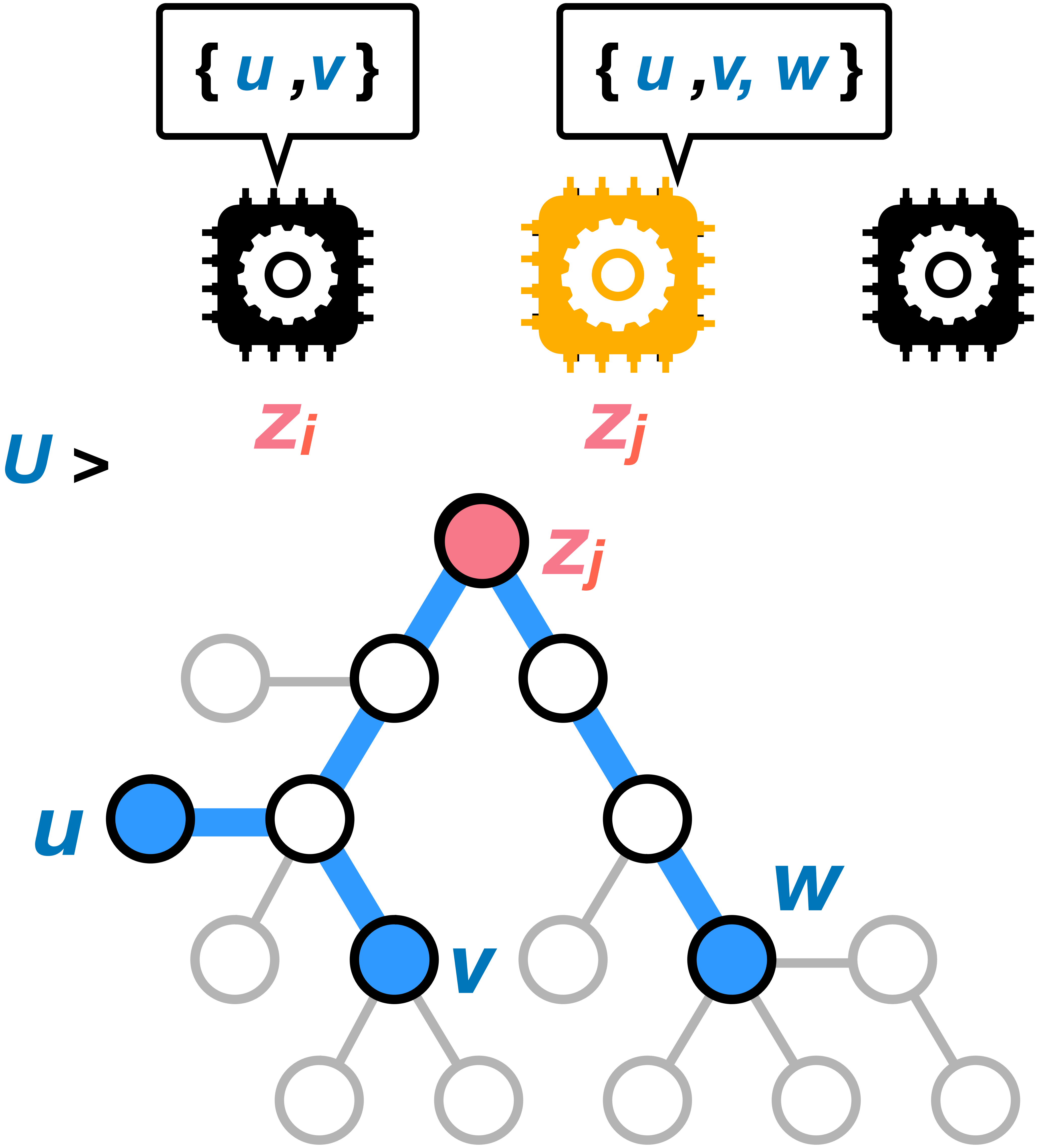
# Algorithm for trees

**Update rule:**

let $g(U)$ to be a vertex in the *center* of $\langle U \rangle$

**Containment property:**

$V_{j(1)} \subseteq V_{j(2)} \subseteq \ldots \subseteq V_{j(n)}$

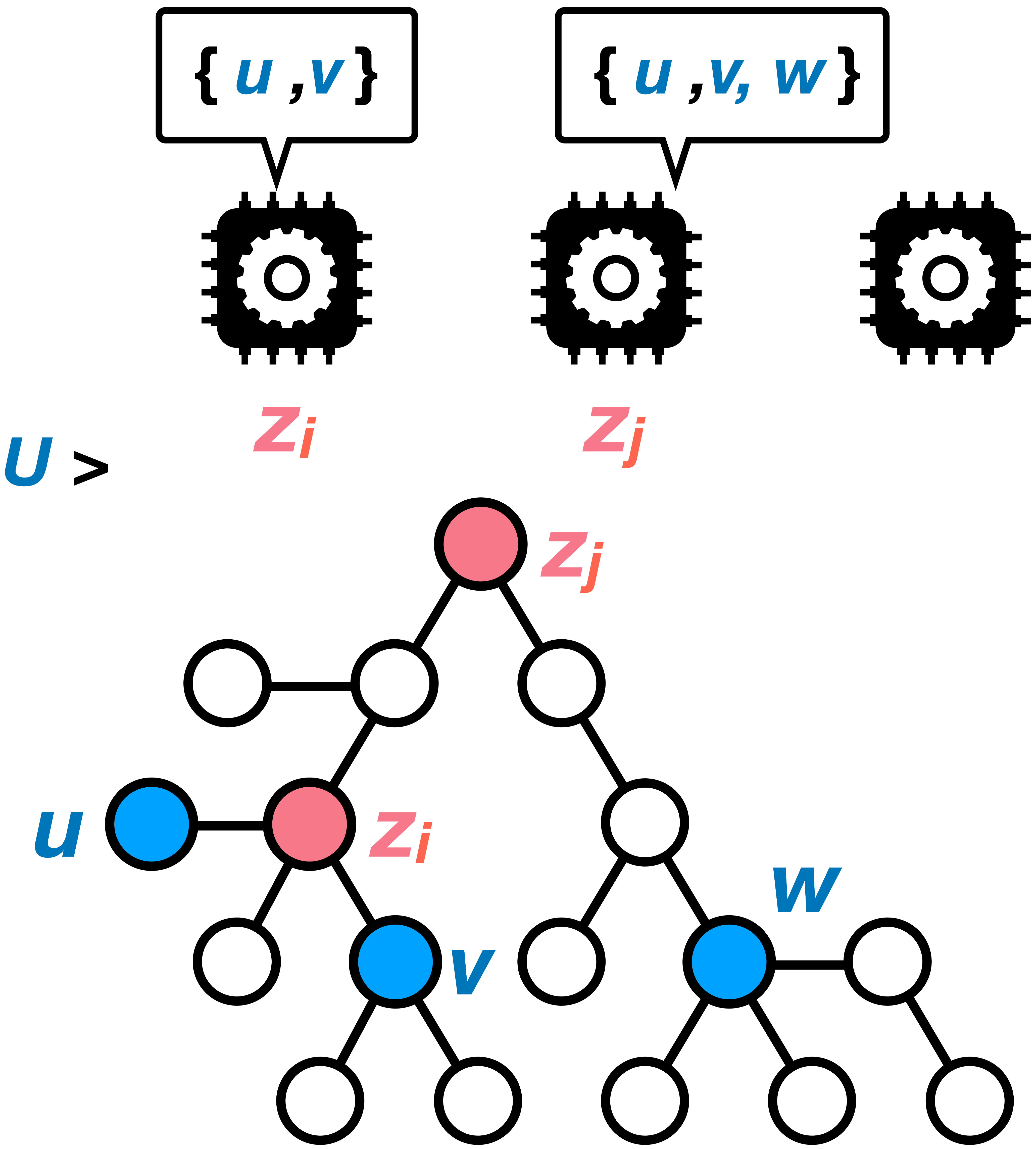


# Algorithm for trees

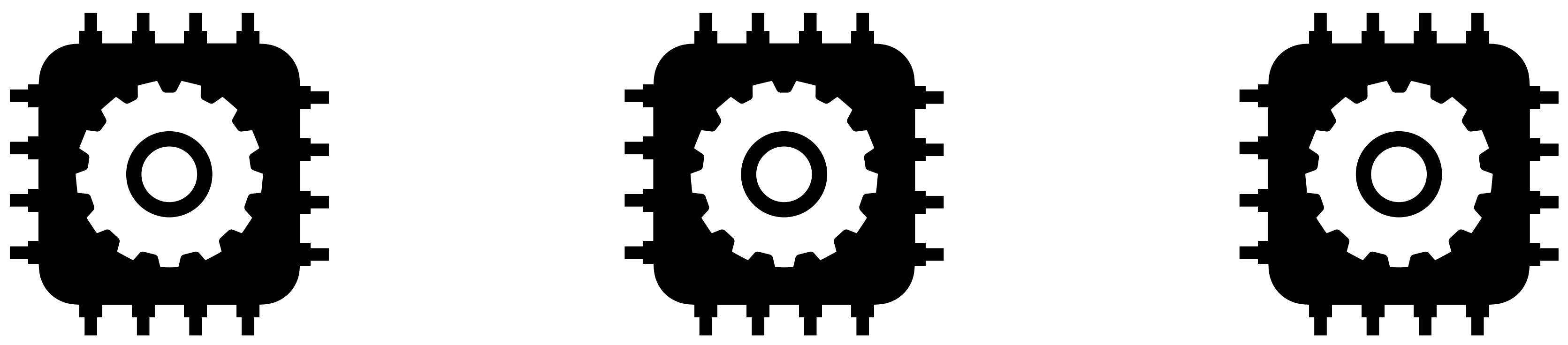
**Update rule:**
let $g(U)$ to be a vertex in the *center* of $< U >$

**Containment property:**
$V_{j(1)} \subseteq V_{j(2)} \subseteq \ldots \subseteq V_{j(n)}$

Distance is now (roughly) half of diameter of $< X >$.



# Algorithm for trees



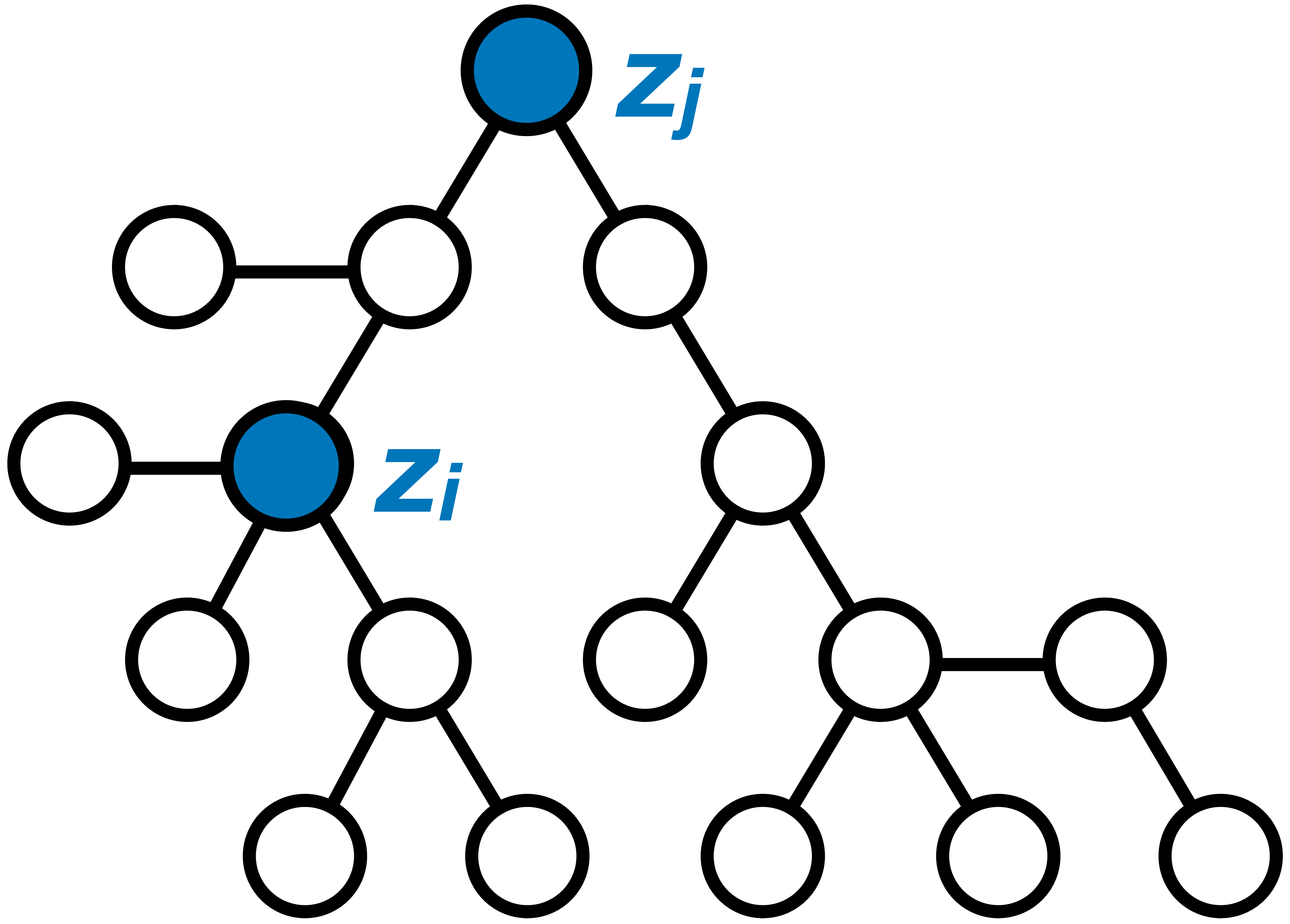
**Update rule:**

let $g(U)$ to be a vertex in the *center* of $< U >$

**Containment property:**

$V_{j(1)} \subseteq V_{j(2)} \subseteq \dots \subseteq V_{j(n)}$

Distance is now (roughly) half of diameter of $< X >$. Repeat with new values as input



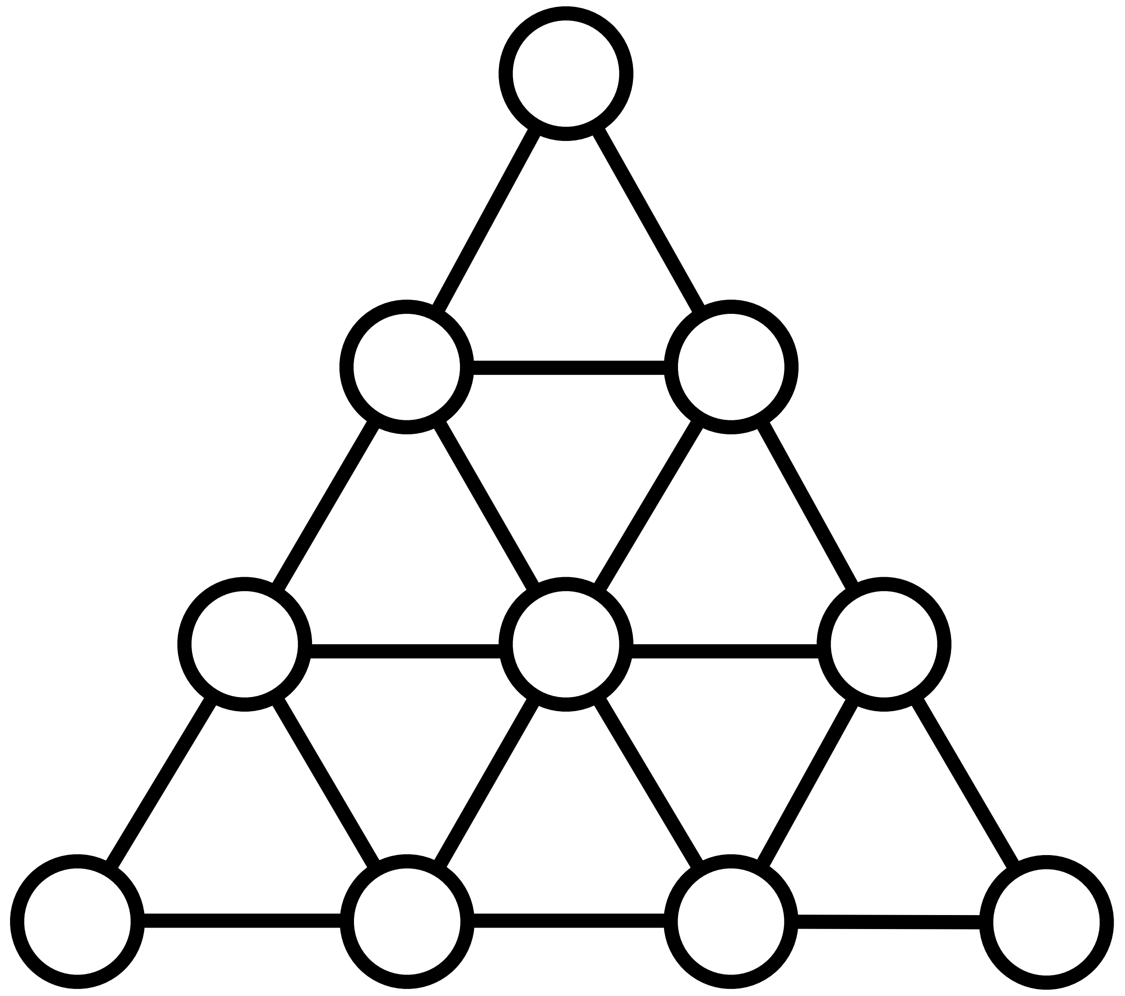
# Algorithm for trees

**Update rule:**
let ***g( U )*** to be a vertex in the *center* of **< *U* >**

**Containment property:**
$V_{j(1)} \subseteq V_{j(2)} \subseteq \ldots \subseteq V_{j(n)}$

Distance is now (roughly) half of diameter
of **< *X* >**. Repeat with new values as input

Same idea extends:
— **chordal graphs:** radius ≈ 1/2 · diameter
— **bridged* graphs:** radius ≈ 2/3 · diameter



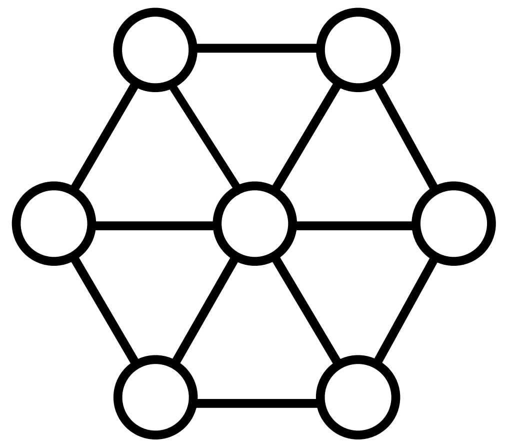
# Wait-free solvability in other graph classes

— **Paths**

e.g, Biran, Moran, Zaks (1990); Attiya, Lynch, Shavit (1994); Schenk (1995)

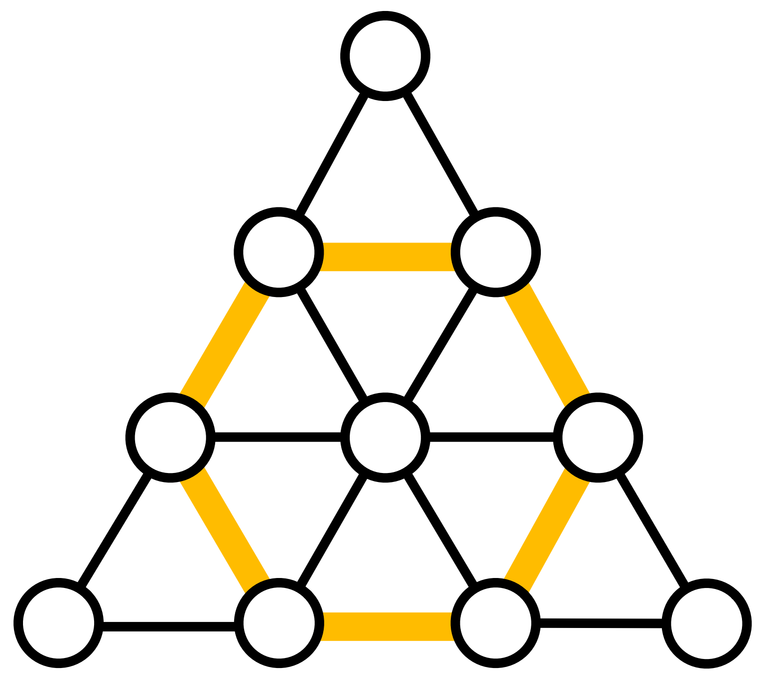
— **Clique graph is a tree or has radius one**

Alcántara, Castañeda, Flores-Peñaloza, Rajsbaum (2019)



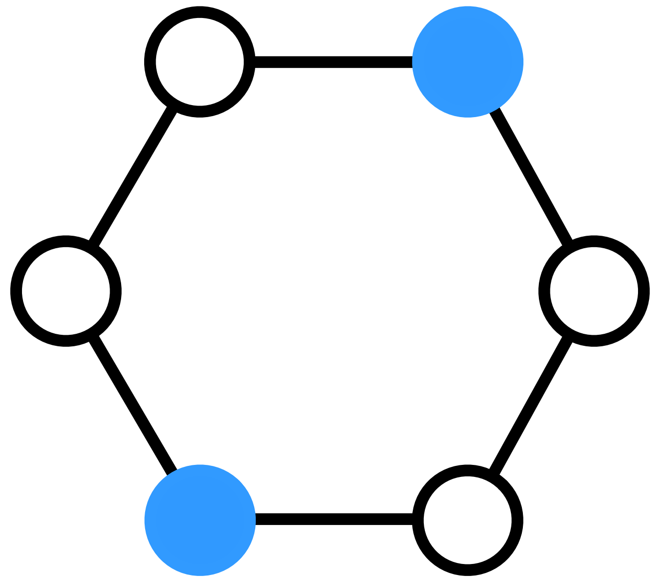
— **Nicely bridged graphs (contains all chordal graphs)**

Alistarh, Ellen, Rybicki (2023)



# **Impossibility results**
# for wait-free algorithms

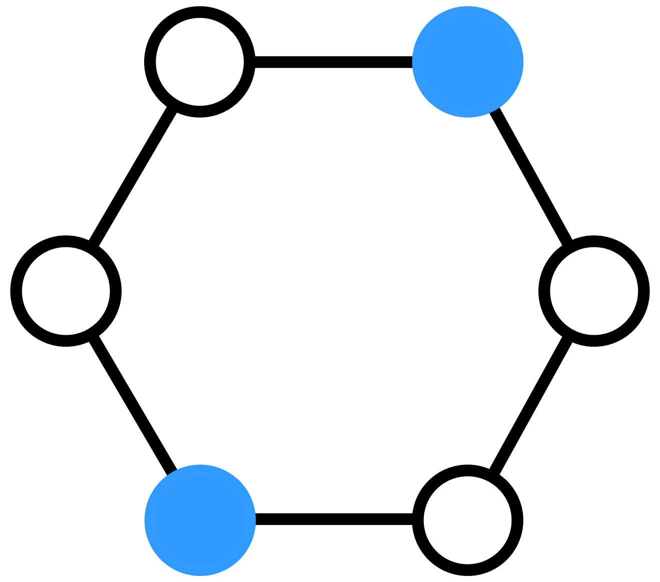
# Impossibility on cycles



**Theorem.** There is **no wait-free algorithm** for ***n* > 2** processes that solves approximate agreement on cycles of length at least 4.

Castañeda, Rajsbaum, and Roy (2018)

# Impossibility on cycles

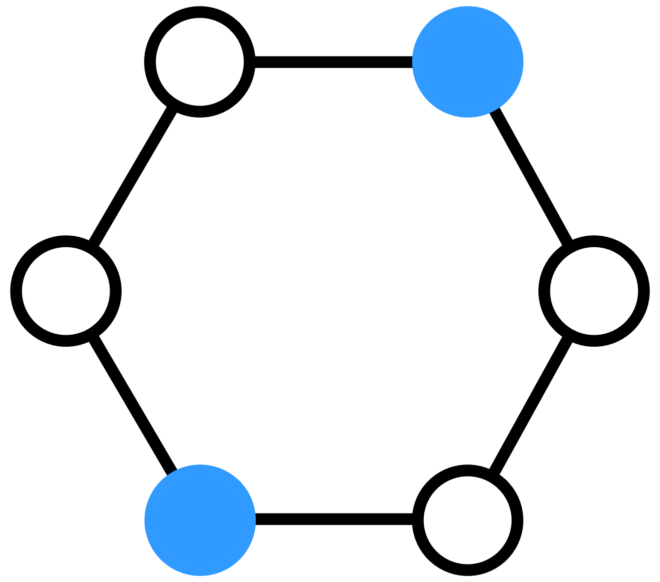


**Theorem.** There is **no wait-free algorithm** for $\boldsymbol{n > 2}$ processes that solves approximate agreement on cycles of length at least 4.

**Corollary.** Any $f$-resilient **synchronous message-passing algorithm** requires at least $\lfloor f/2 \rfloor + 1$ rounds.

**Proof:** Apply BG simulation + Gafni's round-by-round fault-detectors.

# Impossibility on cycles



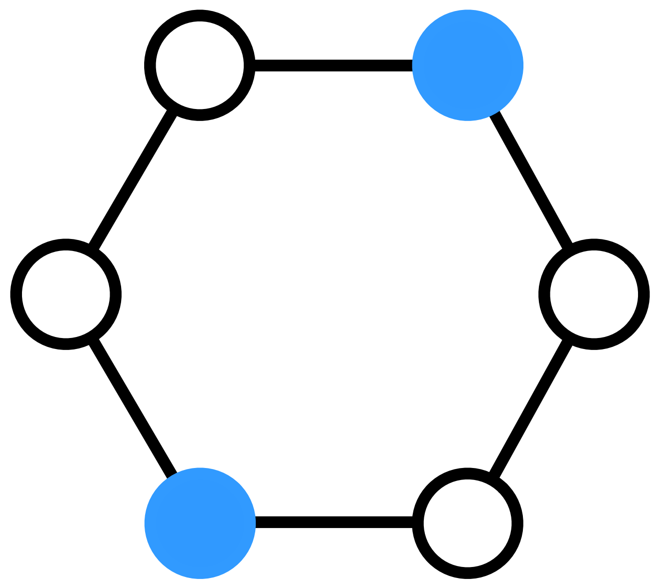
**Theorem.** There is **no wait-free algorithm** for $\boldsymbol{n > 2}$ processes that solves approximate agreement on cycles of length at least 4.

**Two flavours of proofs exist:**

— **Reductions** from 2-set agreement
e.g., Castañeda, Rajsbaum, and Roy (2018)

— **Topological proofs** using variants of Sperner's lemma
e.g., Alistarh, Ellen, Rybicki (2023) and Liu (2022)

# Impossibility on cycles



**Theorem.** There is **no wait-free algorithm** for ***n* > 2** processes that solves approximate agreement on cycles of length at least 4.

**Two flavours of proofs exist:**

— **Reductions** from 2-set agreement
e.g., Castañeda, Rajsbaum, and Roy (2018)

— **Topological proofs** using variants of Sperner's lemma
e.g., Alistarh, Ellen, Rybicki (2023) and Liu (2022)

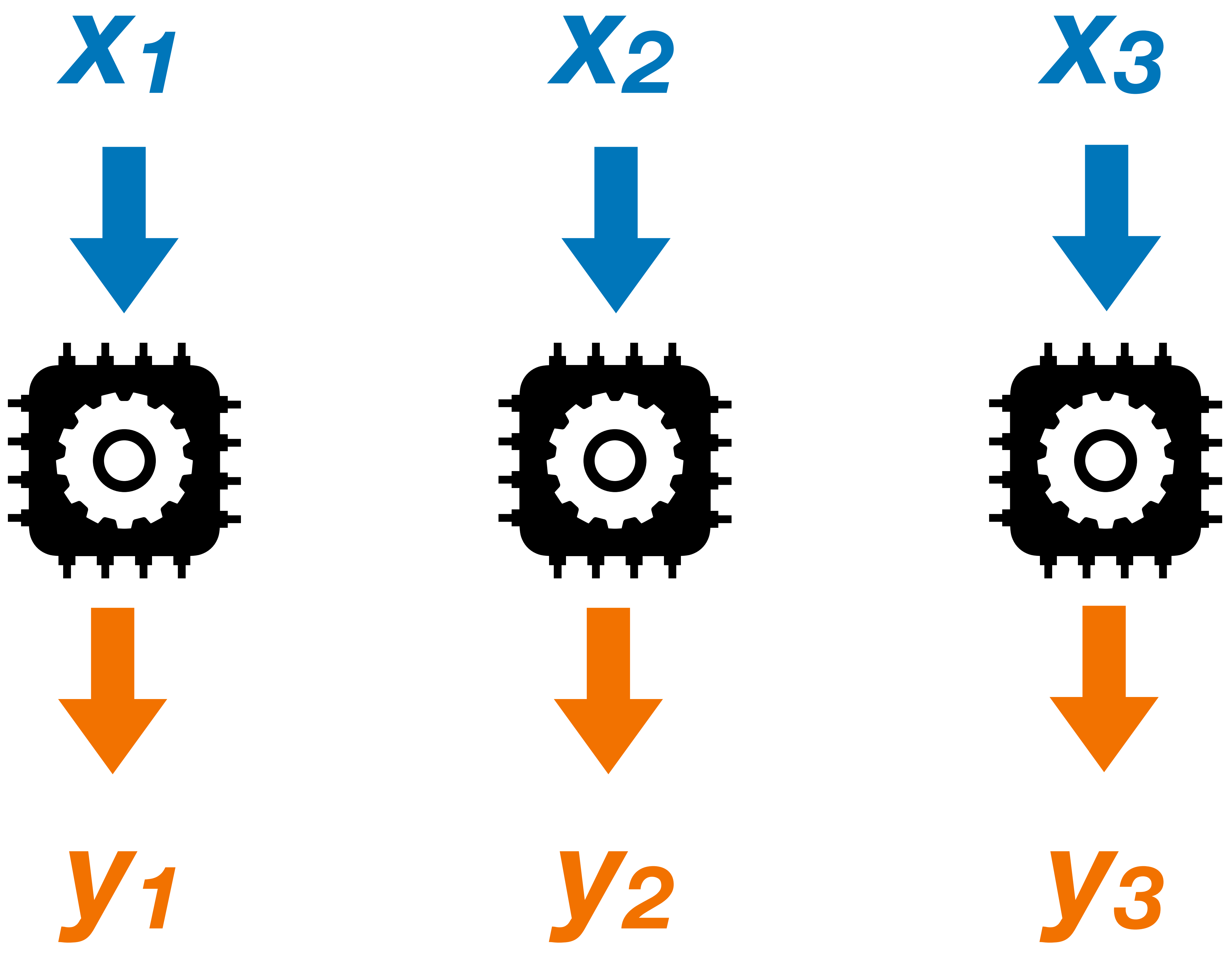
# **A hard problem:** 2-set agreement

$X \subseteq \{ 1, 2, 3 \}$

Constraints:

— **validity:** $Y \subseteq X$

— **agreement:** $| Y | \leq 2$.



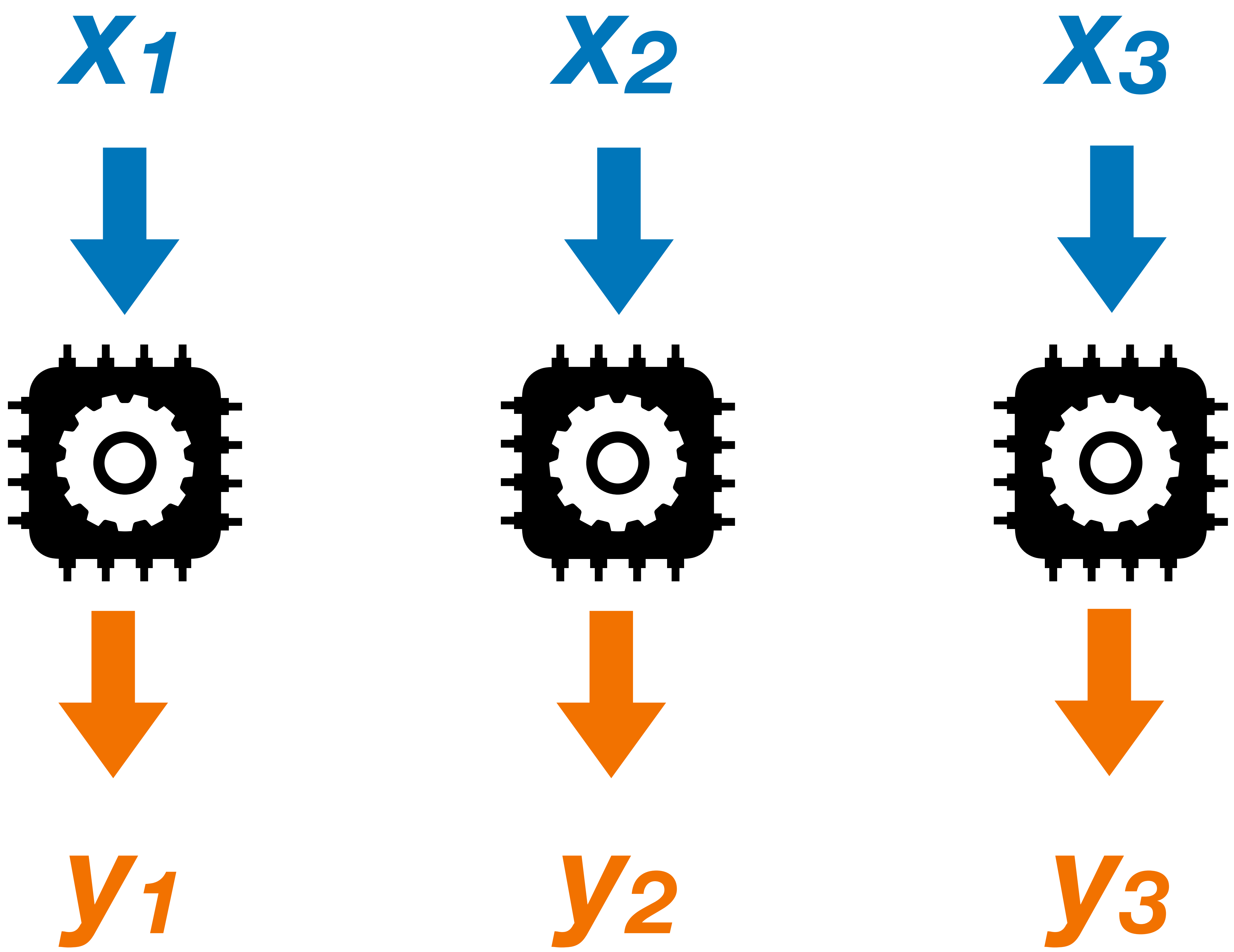
# **A hard problem:**
# 2-set agreement

$X \subseteq \{1, 2, 3\}$

Constraints:

— **validity:** $Y \subseteq X$

— **agreement:** $|Y| \leq 2$.




**Theorem.**

There is no ***wait-free*** algorithm for 2-set agreement for n > 2.

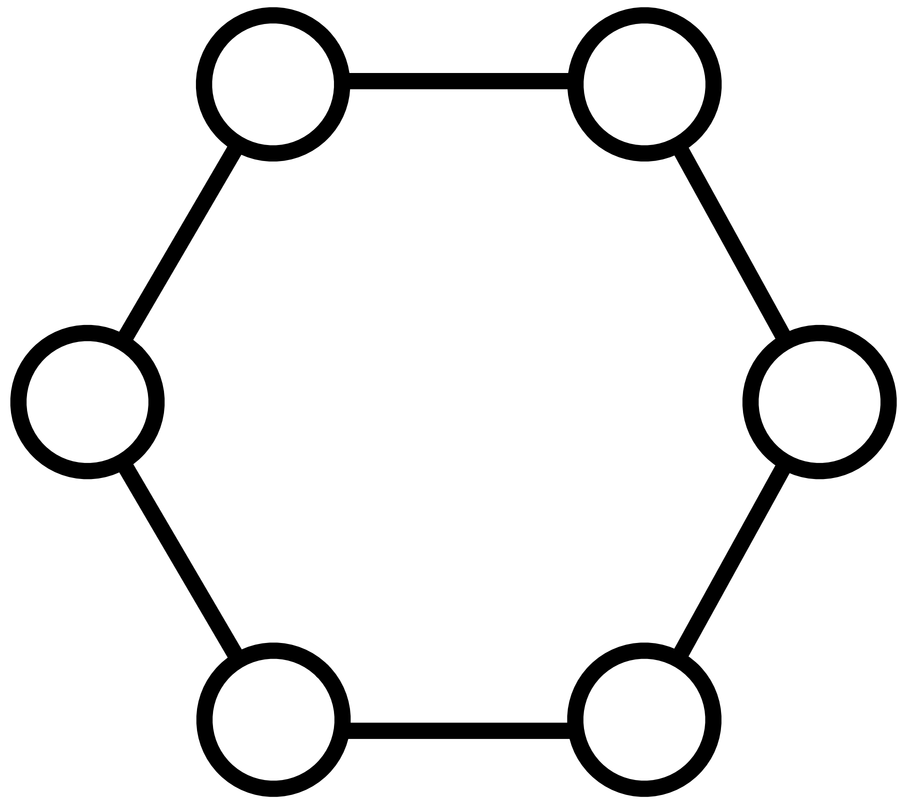
Borowsky and Gafni (1993), Herlihy and Shavit (1999), Saks and Zaharoglou (2000)

# A reduction from 2-set agreement

## Idea:

Suppose there is a wait-free algorithm that solves approximate agreement on ***C***.

Then we can solve 2-set agreement.



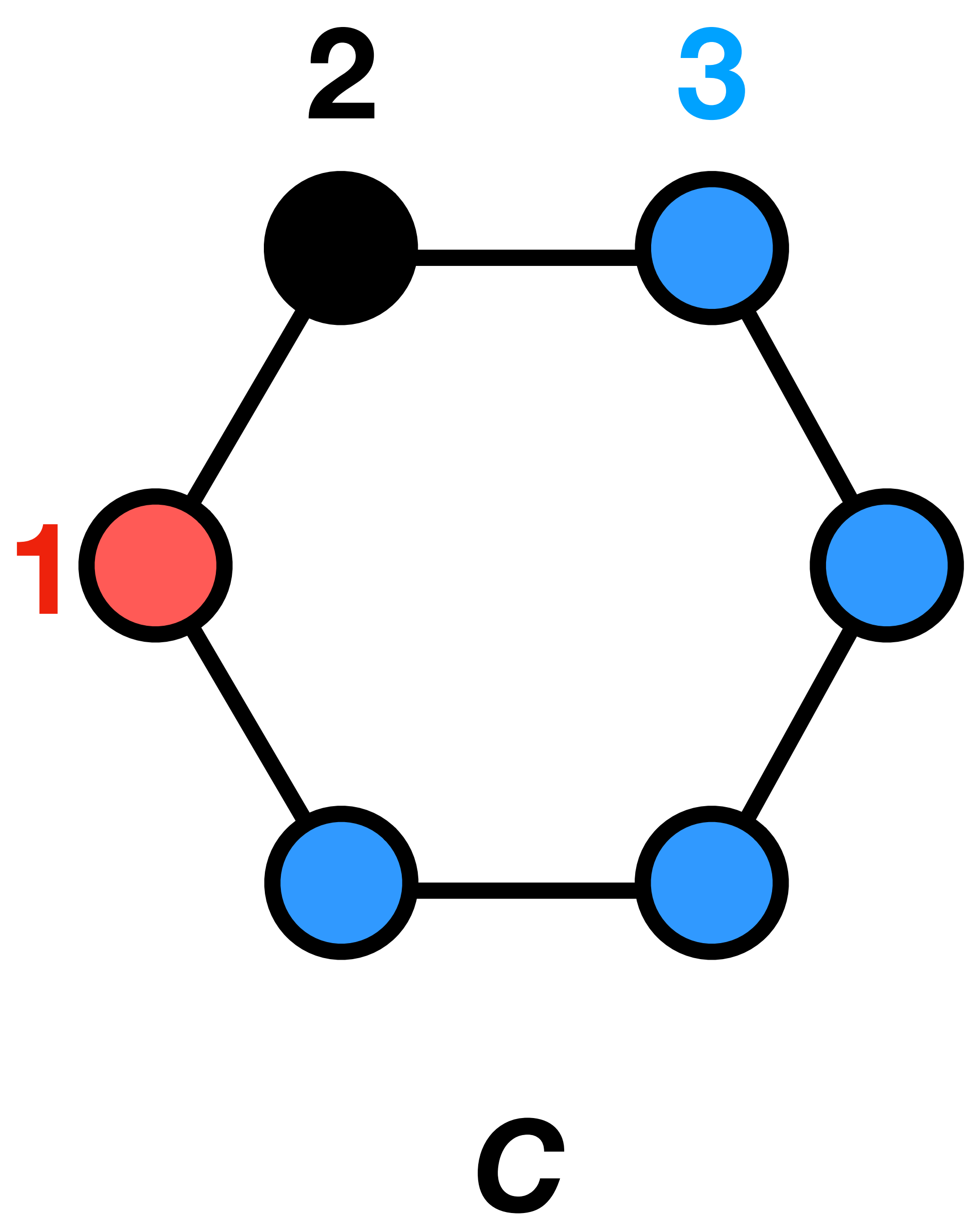
***C***

# A reduction from 2-set agreement

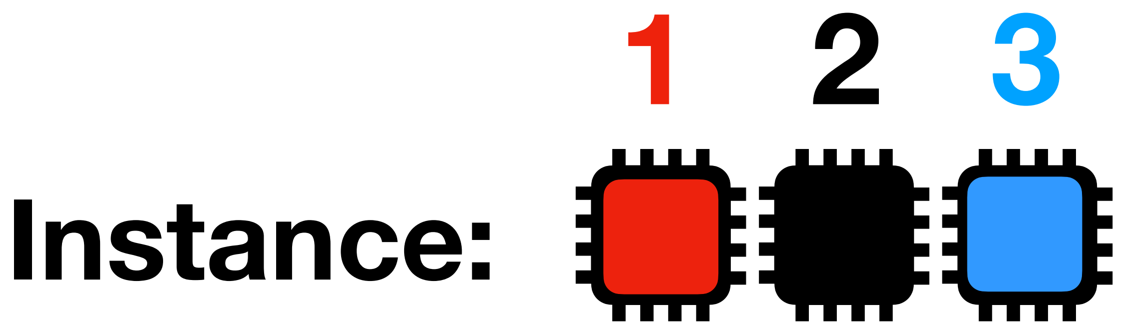
## Idea:

Suppose there is a wait-free algorithm that solves approximate agreement on ***C***.

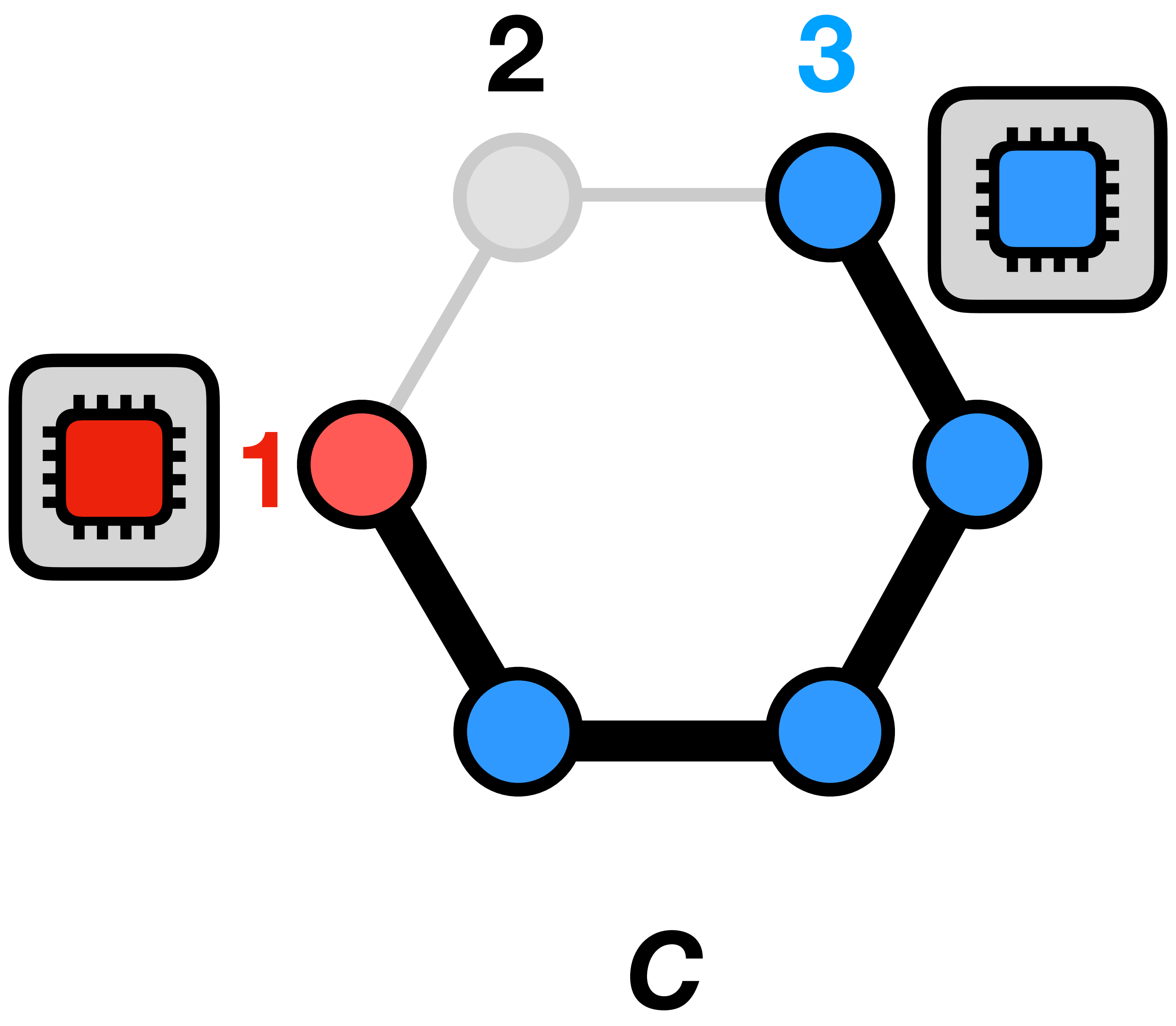
Then we can solve 2-set agreement.



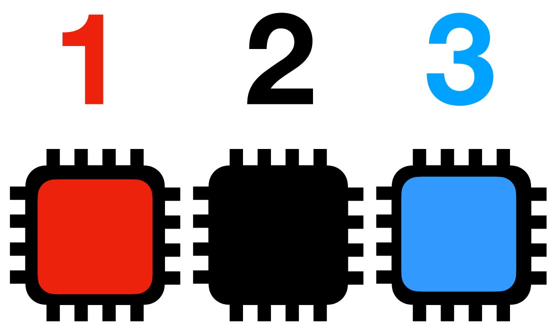
# A reduction from 2-set agreement

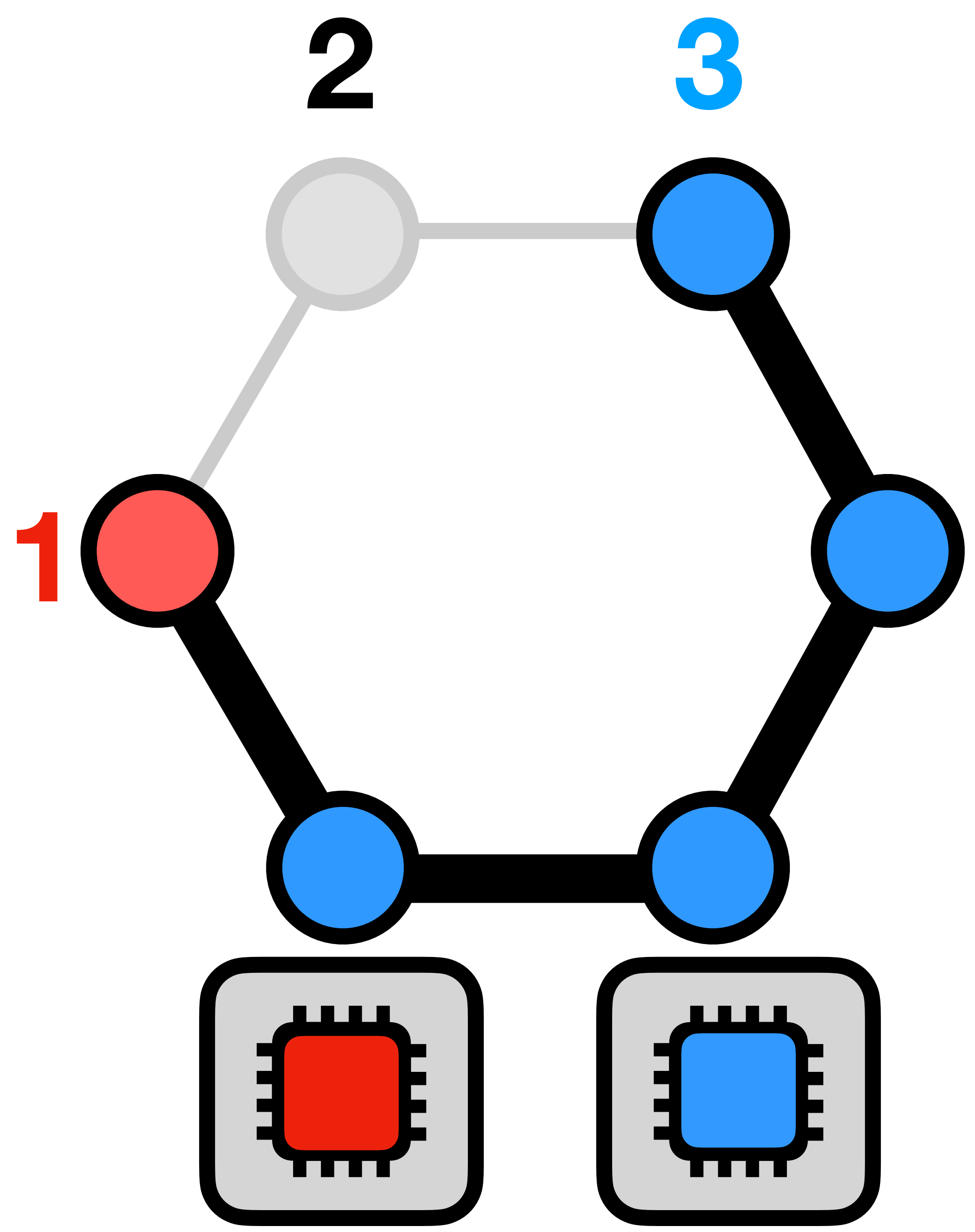



1. Processes with inputs **{ 1, 3 }** run a wait-free algorithm for approximate agreement on the **path** obtained by removing the black vertex **2**.



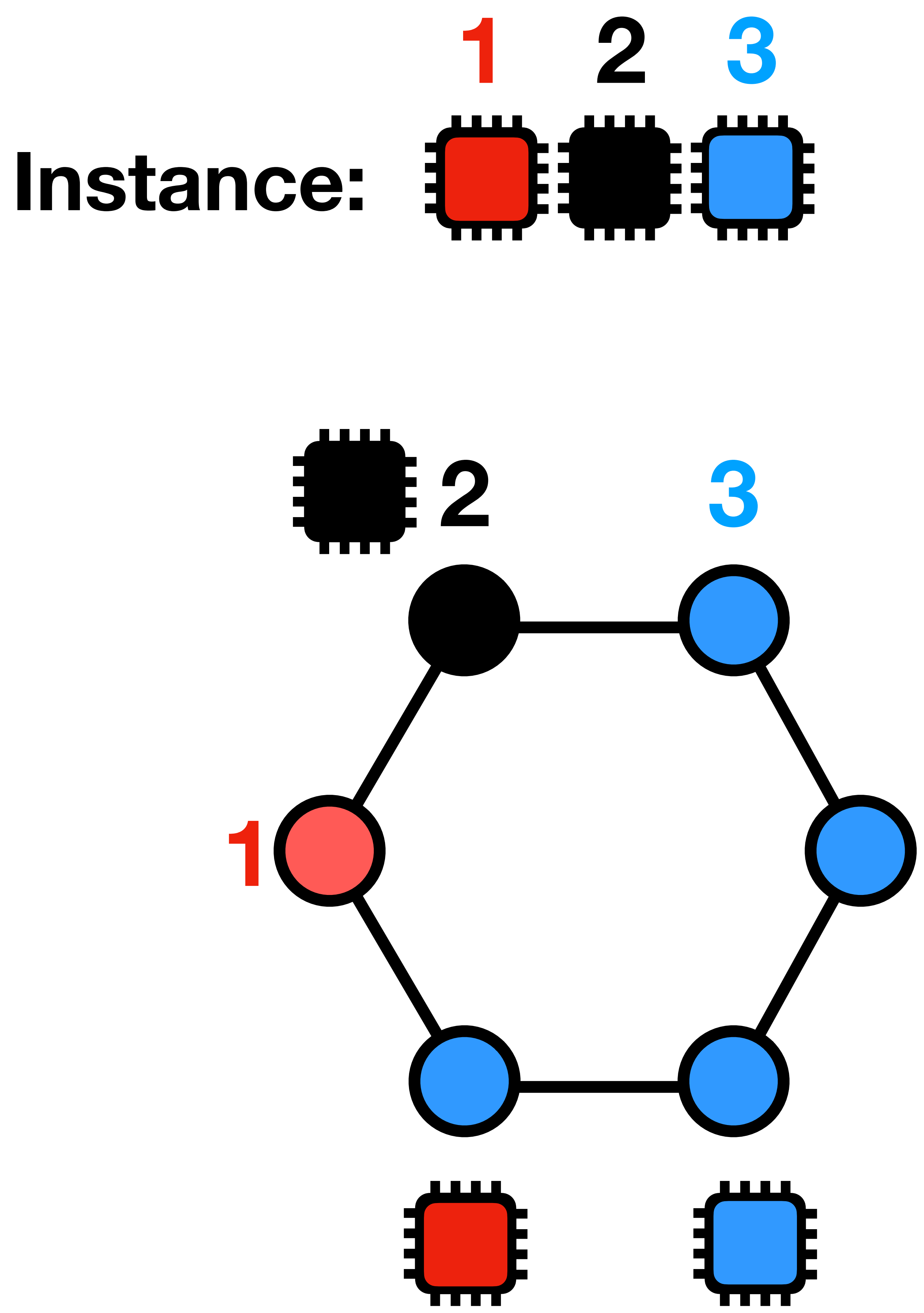
# A reduction from 2-set agreement

Instance: 


1. Processes with inputs **{** 1, 3 **}** run a wait-free algorithm for approximate agreement on the **path** obtained by removing the black vertex **2**.

# A reduction from 2-set agreement

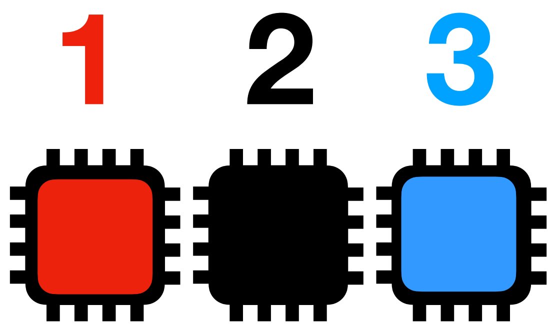


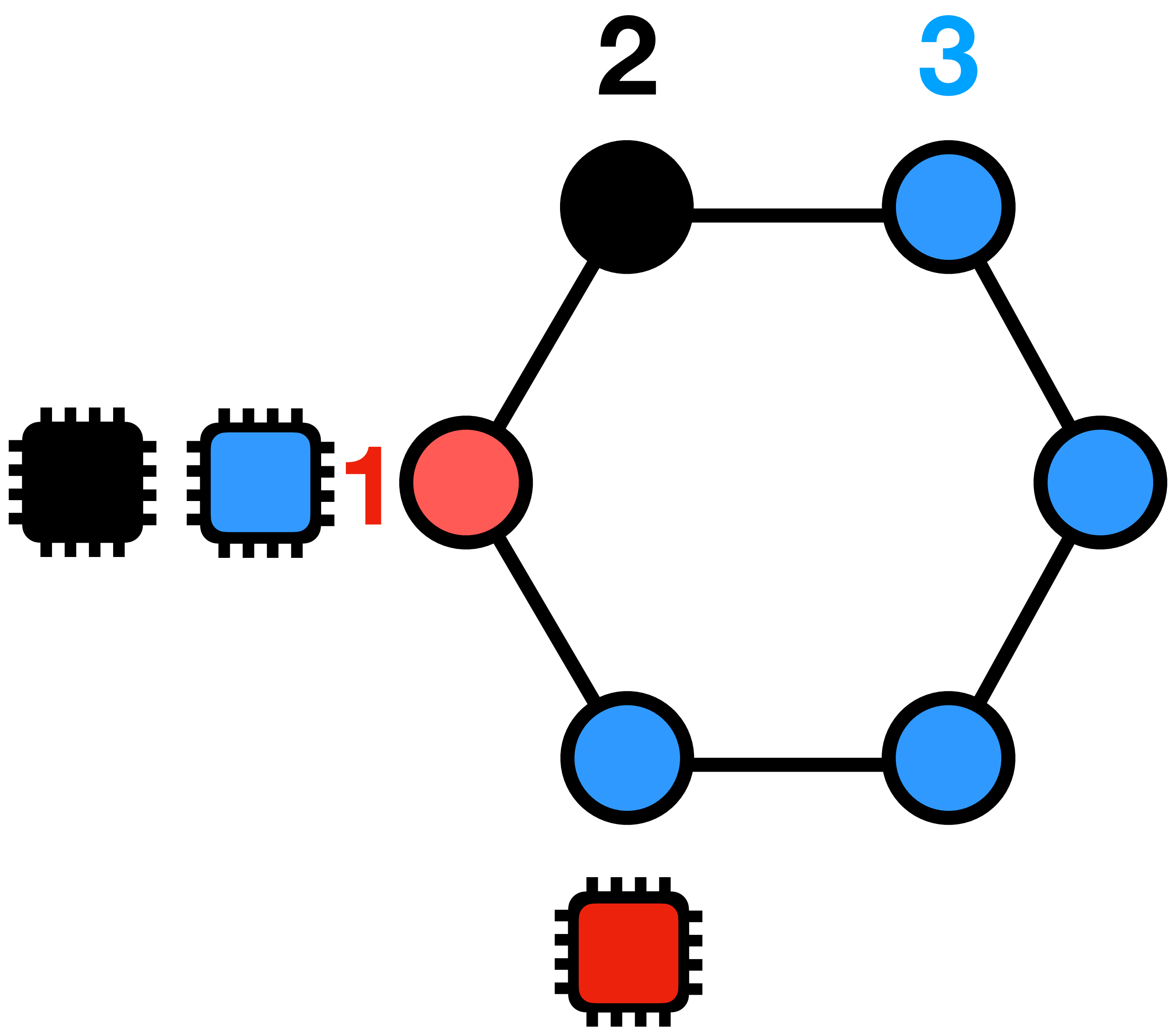

1. Processes with inputs **{ 1, 3 }** run a wait-free algorithm for approximate agreement on the **path** obtained by removing the black vertex **2**.

2. Then **all** processes run the approximate agreement protocol on the cycle.

# A reduction from 2-set agreement

Instance:

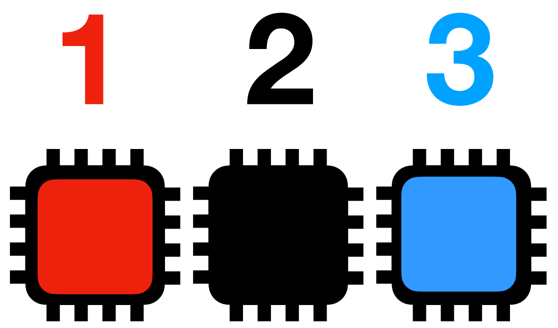


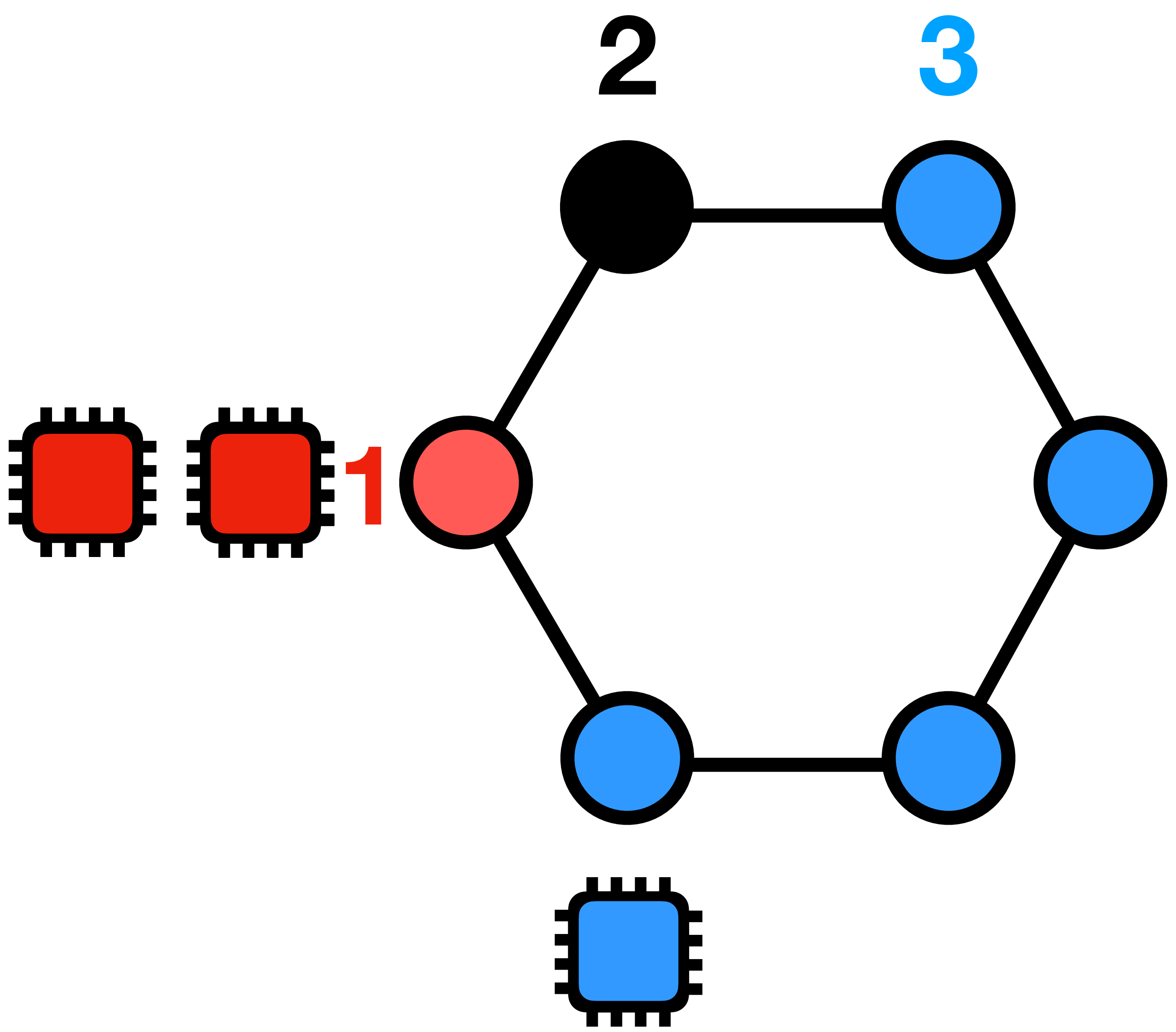




1. Processes with inputs **{** 1, 3 **}** run a wait-free algorithm for approximate agreement on the **path** obtained by removing the black vertex **2**.

2. Then **all** processes run the approximate agreement protocol on the cycle.

# A reduction from 2-set agreement

Instance: 


1. Processes with inputs **{** 1**,** 3 **}** run a wait-free algorithm for approximate agreement on the **path** obtained by removing the black vertex **2**.

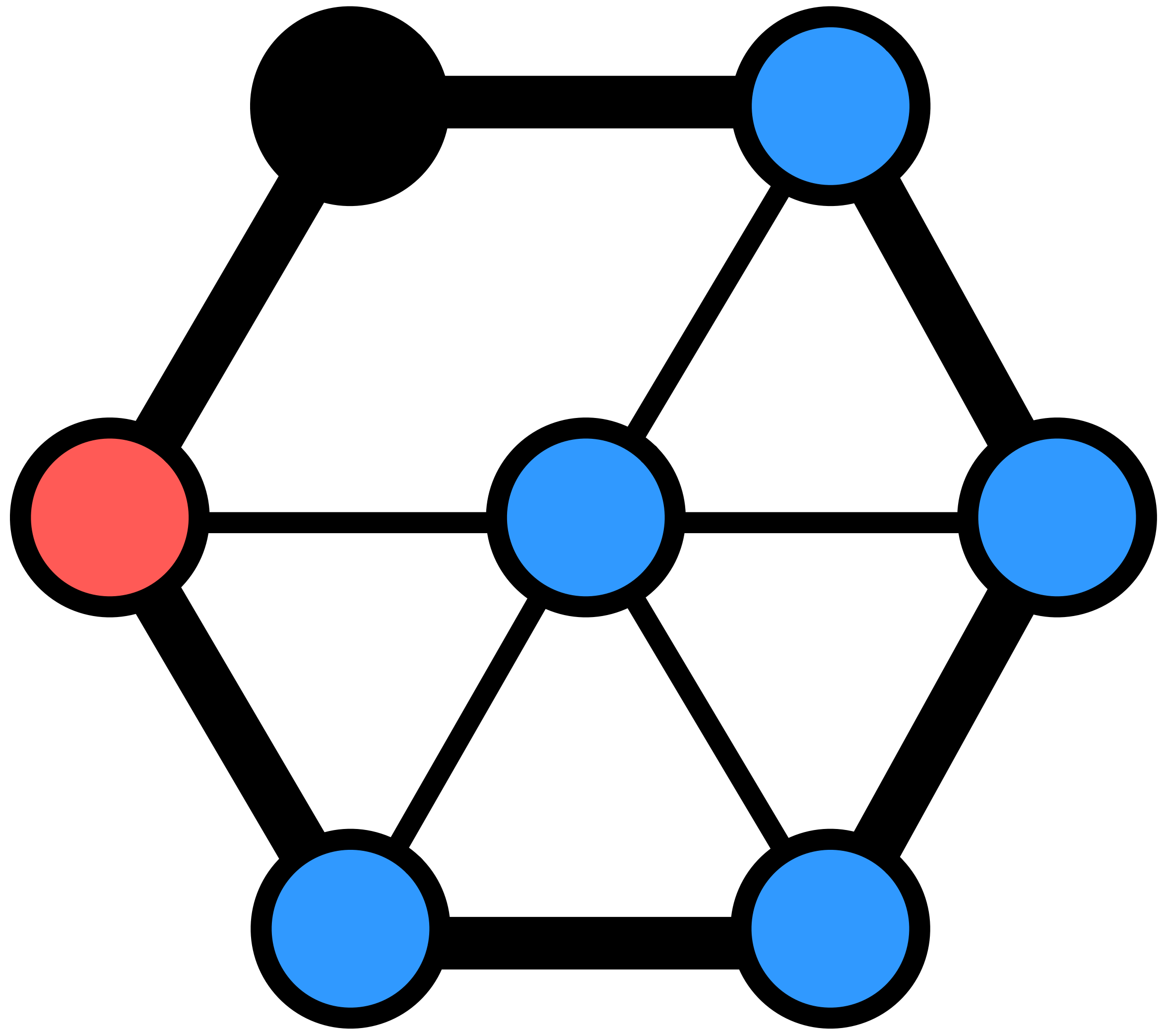
2. Then **all** processes run the approximate agreement protocol on the cycle.

3. Output the ***colour*** of the output vertex.

# **A reduction from 2-set agreement:** beyond cycles

Let $\boldsymbol{L} : \boldsymbol{V} \rightarrow \{\ 1, 2, 3\ \}$ such that

1. there is no triangle with all three colours,
2. there is a cycle ***C*** with three consecutive vertices of colour **1, 2, 3**
3. there is exactly one black node on ***C***



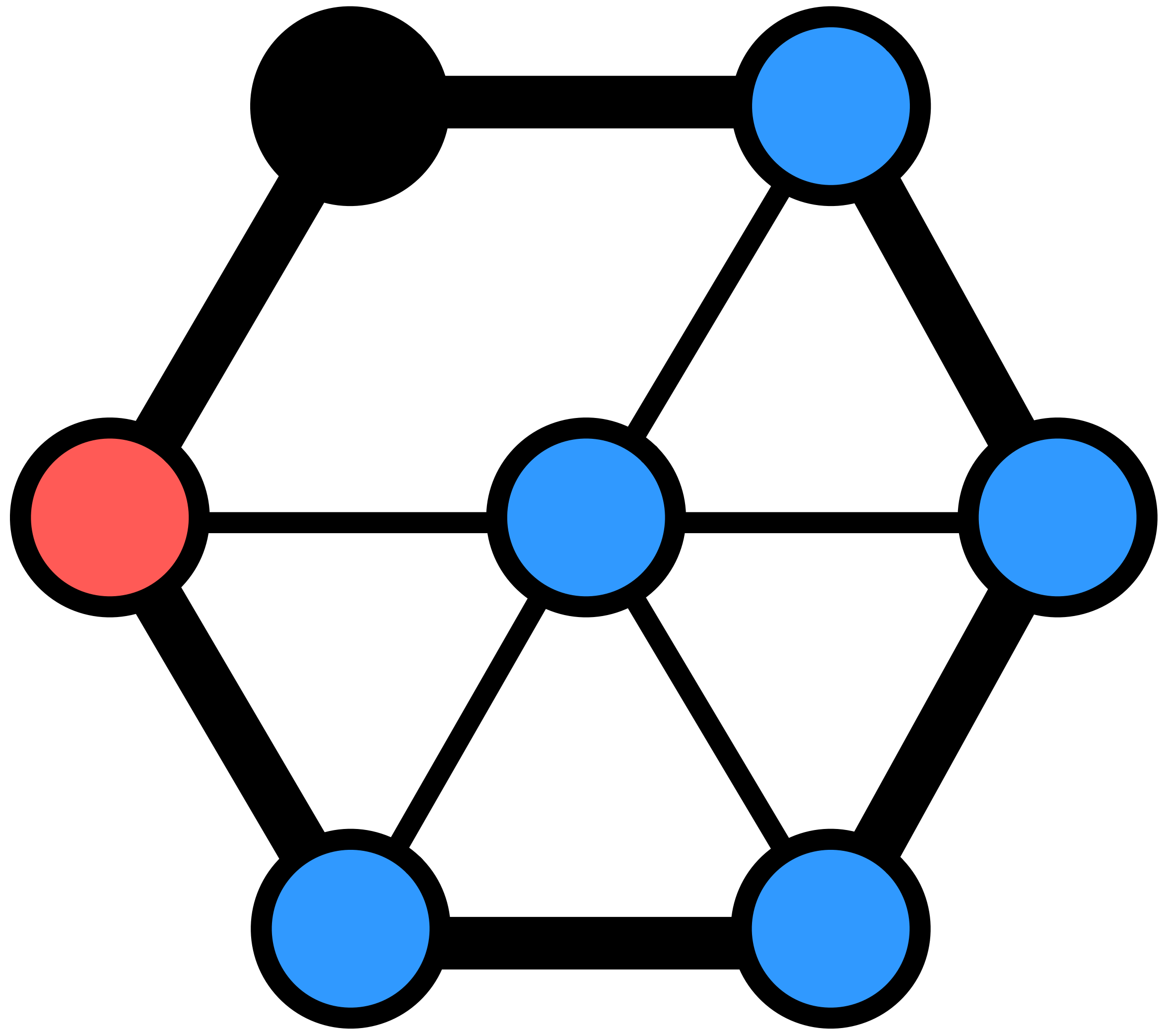
# **A reduction from 2-set agreement:** beyond cycles

Let $L : V \rightarrow \{ 1, 2, 3 \}$ such that

1. there is no triangle with all three colours,
2. there is a cycle $C$ with three consecutive vertices of colour **1, 2, 3**
3. there is exactly one black node on $C$

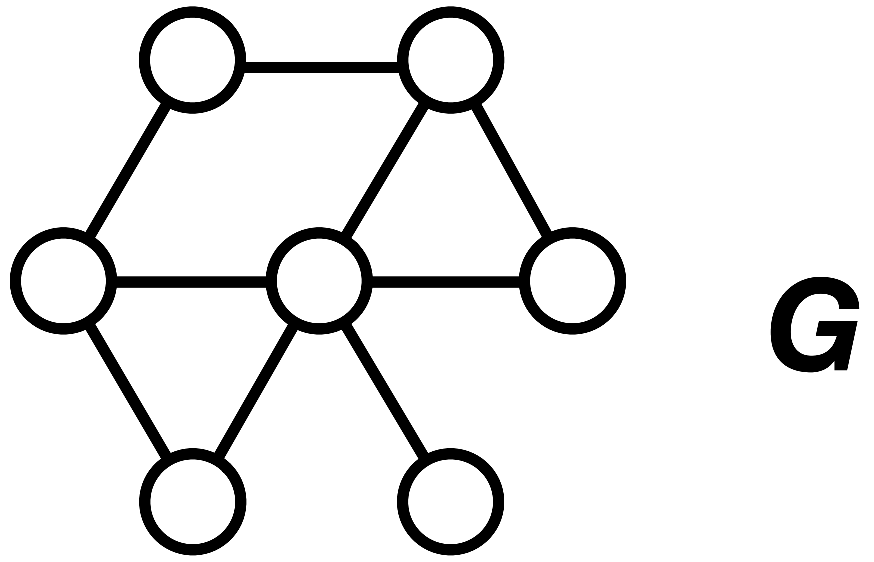
**Theorem (Alistarh, Ellen, Rybicki 2023).**
If $G$ has such an ***impossibility labelling***, then there is no wait-free algorithm for approximate agreement on $G$

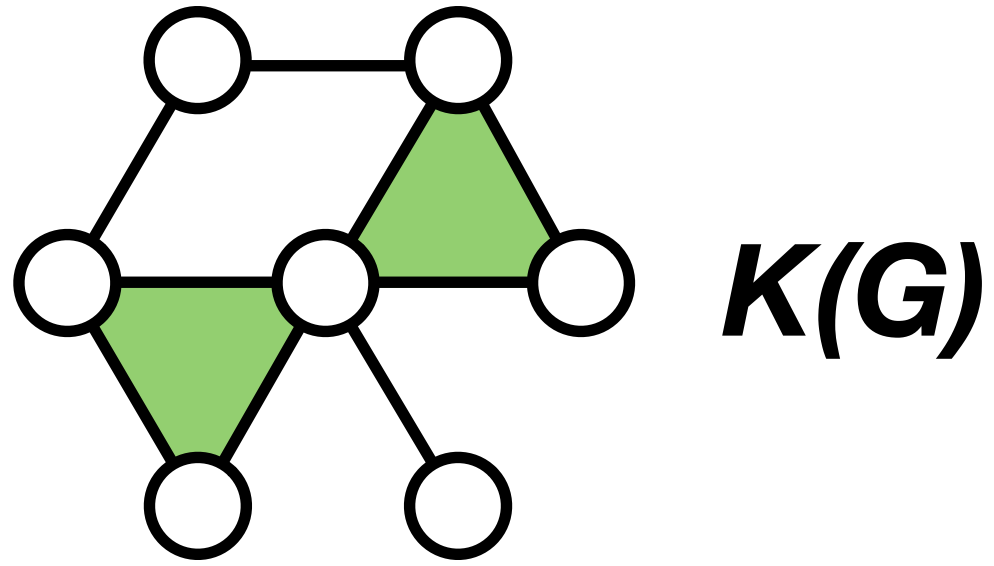
Holds even under **clique validity**.



# Ledent's conjecture

The *complex of cliques of* ***G*** is the complex ***K(G)*** = (***V***, **S**), where **S** is the set of all cliques of ***G*.**

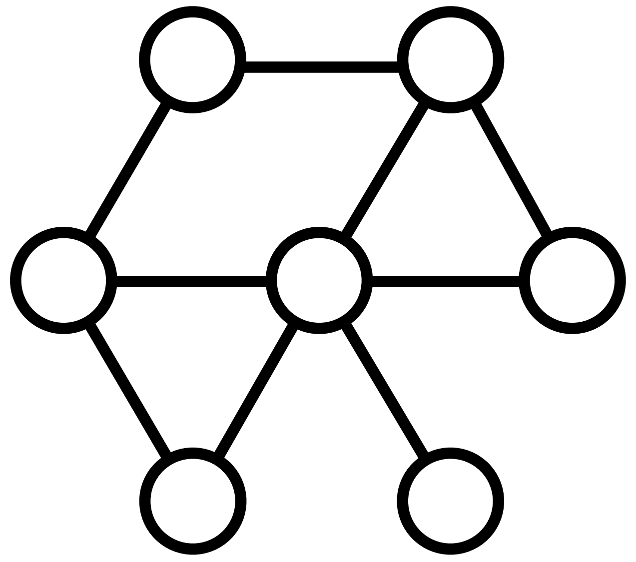


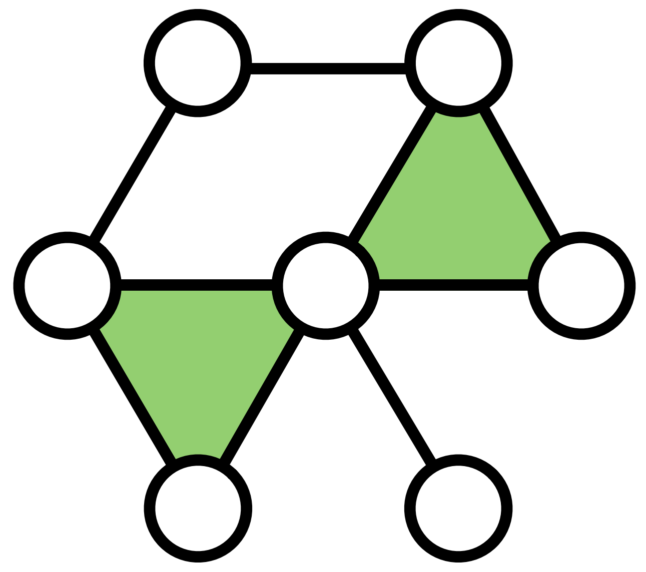

**Ledent's conjecture (PODC 2021):**
Approximate agreement under clique validity is wait-free solvable on ***G*** if and only if ***K(G)*** is contractible**.**

# Ledent's conjecture

The *complex of cliques of* ***G*** is the complex ***K(G)*** = (***V***, **S**), where **S** is the set of all cliques of ***G*.**

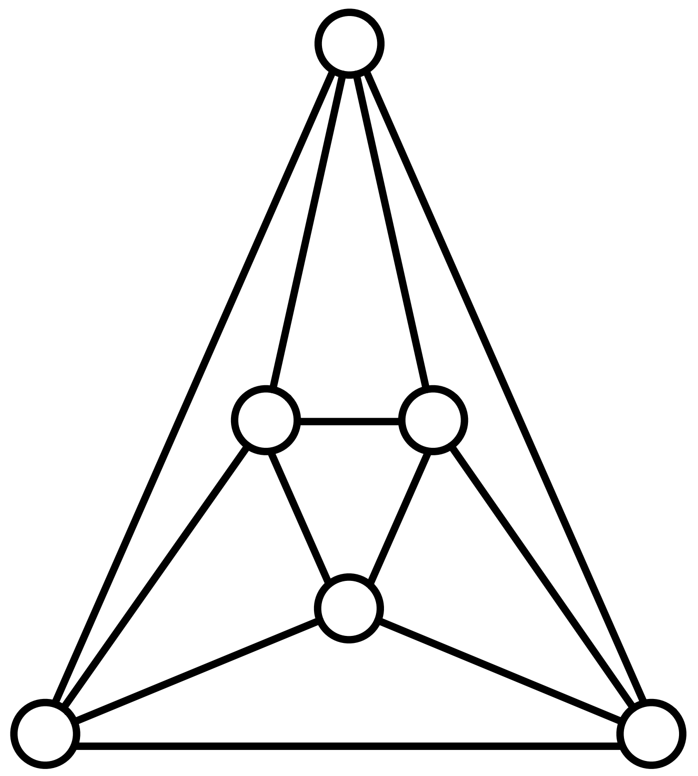



**Ledent's conjecture (PODC 2021):**
Approximate agreement under clique validity is wait-free solvable on ***G*** if and only if ***K(G)*** is contractible**.**



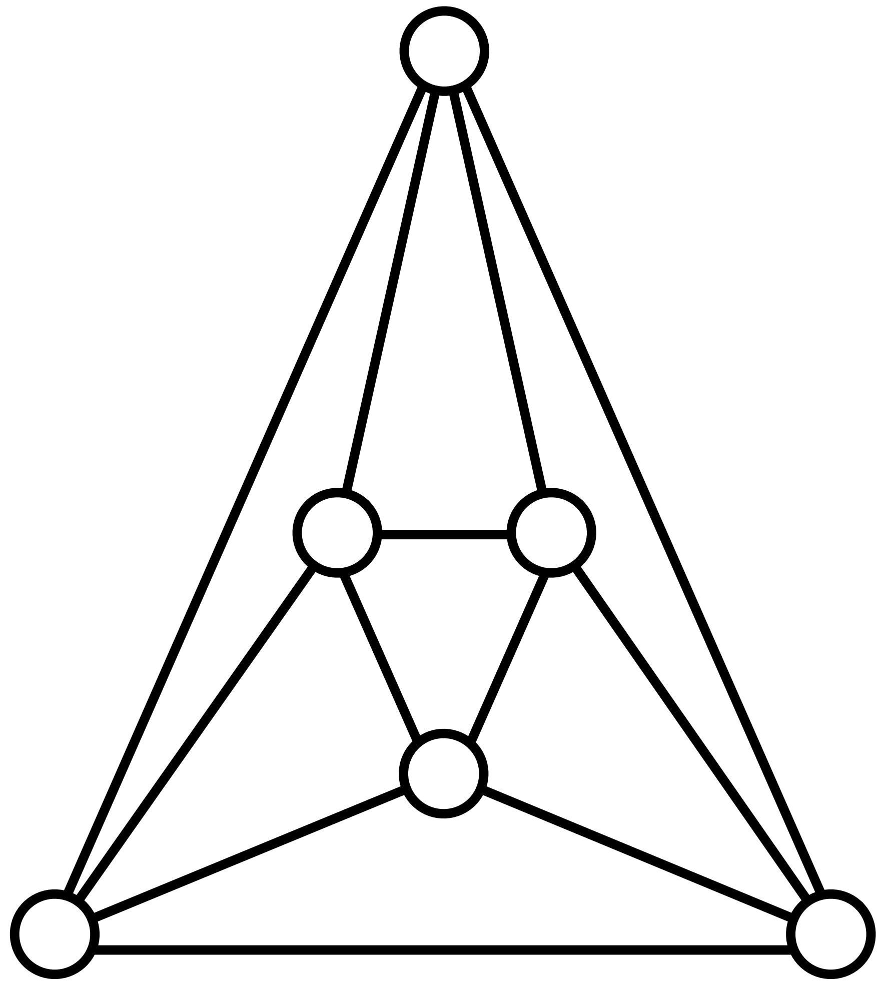

**Interesting case:** *triangulated spheres*
— non-contractible complex of cliques
— no impossibility labelling

Are there wait-free algorithms for such graphs?



octahedron

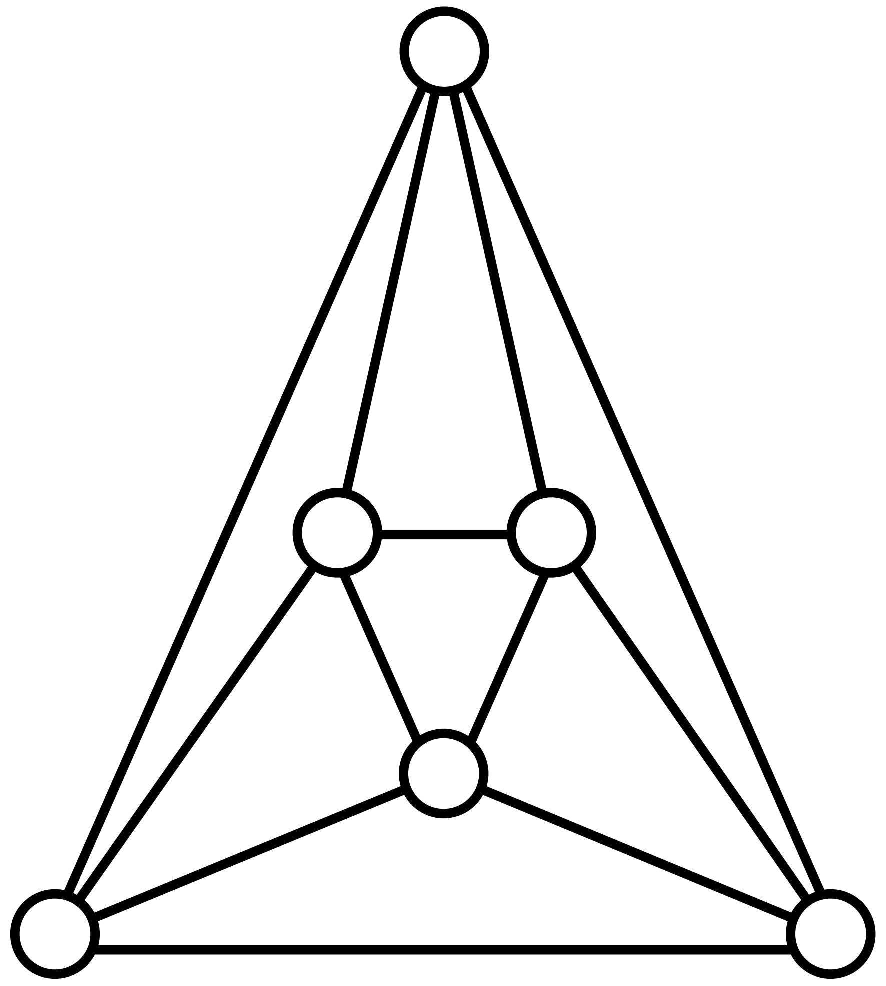
# Liu’s theorem



octahedron

**Liu (2022):**
Octahedron does **not** have an impossibility labelling
and does **not** have a wait-free algorithm for ***n*>3** processes!

# Liu's theorem



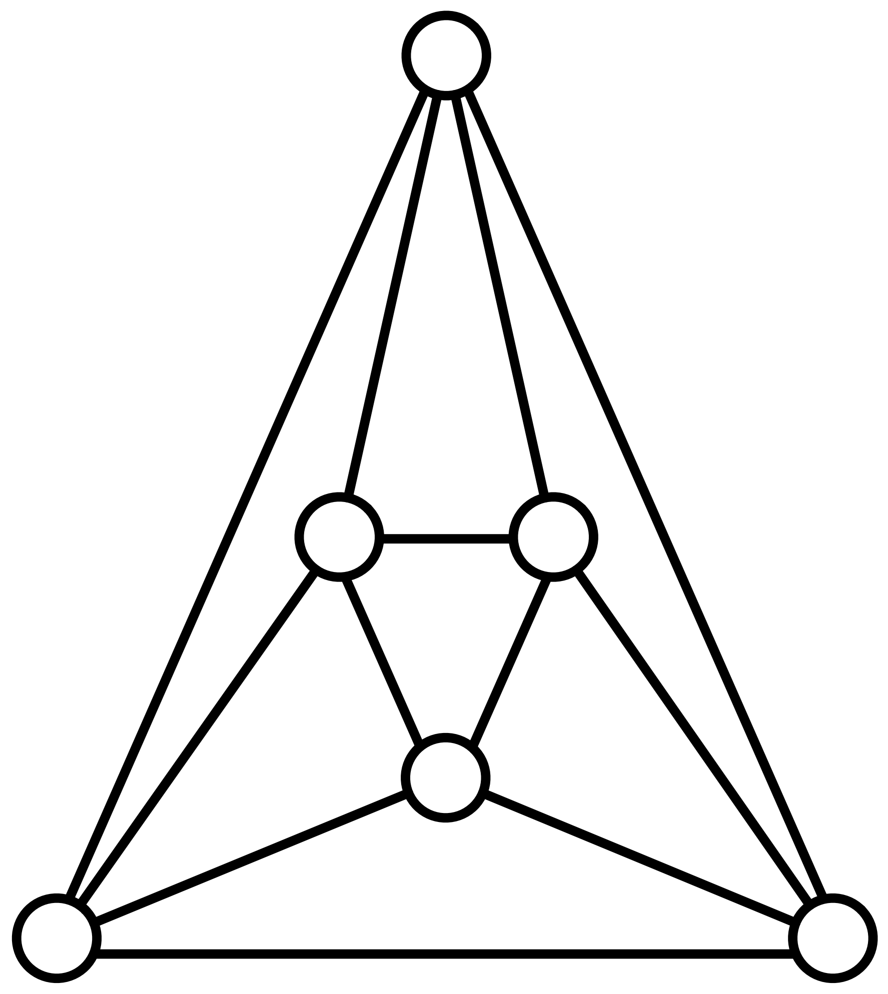
octahedron

**Liu (2022):**
Octahedron does **not** have an impossibility labelling
and does **not** have a wait-free algorithm for ***n*>3** processes!

**Liu's theorem:**
If ***G*** satisfies a ***k-clique containment condition***,
then there is no wait-free protocol for ***n* > *χ(G)*** processes.

***χ(G):*** chromatic number of ***G***

# Liu's theorem



octahedron

**Liu (2022):**
Octahedron does **not** have an impossibility labelling
and does **not** have a wait-free algorithm for ***n*>3** processes!

**Liu's theorem:**
If ***G*** satisfies a ***k-clique containment condition***,
then there is no wait-free protocol for ***n* > *χ(G)*** processes.

For example, **triangulated *d*-dimensional spheres**
satisfy the (*d+1*)-clique containment condition.

**Open problem:** Is there a matching upper bound?

# Summary

## Upper bound techniques

- "iterative **pruning** of convex hull", works in chordal graphs and "nicely bridged" graphs: Alistarh et al. (2023)

# Summary

## Upper bound techniques

- “iterative **pruning** of convex hull”, works in chordal graphs and “nicely bridged” graphs: Alistarh et al. (2023)

## Lower bound techniques

- **reductions:** Castañeda et al. (2018), Alcántara et al. (2019), Alistarh et al. (2023), Liu (2022)
- **topological proofs:** Alistarh et al. (2023), Liu (2022)

# What else is there?

## Extension-based proofs

- **no “simple” impossibility proofs exist:** e.g., Alistarh et al. (2021), Liu (2022), …

# What else is there?

## Extension-based proofs

- **no “simple” impossibility proofs exist:** e.g., Alistarh et al. (2021), Liu (2022), …

## Message-passing systems with arbitrary faults

- **agreement under minimal paths validity:** Nowak and Rybicki (2019)
- **“best-of-both-worlds”:** Constantinescu, Ghinea, Wattenhofer, Westermann (2023)

# What else is there?

## Extension-based proofs

- **no "simple" impossibility proofs exist:** e.g., Alistarh et al. (2021), Liu (2022), …

## Message-passing systems with arbitrary faults

- **agreement under minimal paths validity:** Nowak and Rybicki (2019)
- **"best-of-both-worlds":** Constantinescu, Ghinea, Wattenhofer, Westermann (2023)

## Connections to other agreement problems

- **robot gathering:** Castañeda et al. (2018), Alcántara et al. (2019)
- **simplex agreement:** Ledent (2021)
- **multi-valued consensus:** Attiya and Welch (2023)

# Open problems

## Open problem 1

- Characterise the class of graphs which admit wait-free algorithms for $n > 2$.

# Open problems

## Open problem 1

- Characterise the class of graphs which admit wait-free algorithms for $n > 2$.

## Open problem 2

- Is there for any $k > 1$ some graph $\boldsymbol{G_k}$ in which approximate agreement is wait-free solvable if and only if $n > k$?

# Open problems

## Open problem 1

- Characterise the class of graphs which admit wait-free algorithms for $n > 2$.

## Open problem 2

- Is there for any $k > 1$ some graph $\boldsymbol{G_k}$ in which approximate agreement is wait-free solvable if and only if $n > k$?

## Open problem 3

- Are there graphs that do not admit algorithms under shortest path validity but admit algorithms under with minimal path/clique validity?

# Open problems

## Open problem 1

- Characterise the class of graphs which admit wait-free algorithms for $n > 2$.

## Open problem 2

- Is there for any $k > 1$ some graph $\boldsymbol{G_k}$ in which approximate agreement is wait-free solvable if and only if $n > k$?

## Open problem 3

- Are there graphs that do not admit algorithms under shortest path validity but admit algorithms under with minimal path/clique validity?

**Thank you!**

# References


- Abraham, Amit, Dolev (OPODIS 2004). https://doi.org/10.1007/11516798_17
- Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum (Distrib. Comput. 2019). https://doi.org/10.1007/s00446-018-0345-3
- Alistarh, Ellen, Rybicki (TCS 2023). https://doi.org/10.1016/j.tcs.2023.113733
- Attiya and Welch (DISC 2023). https://doi.org/10.4230/LIPIcs.DISC.2023.36
- Attiya, Lynch, Shavit (JACM 1994). https://doi.org/10.1145/179812.179902
- Biran, Moran, Zaks (JALG 1990). https://doi.org/10.1016/0196-6774(90)90020-F
- Borowsky and Gafni (STOC 1993). https://doi.org/10.1145/167088.167119
- Castañeda, Rajsbaum, and Roy (J Braz. Comput. Soc. 2018). https://doi.org/10.1186/s13173-017-0065-8
- Constantinescu, Ghinea, Wattenhofer, Westermann (Cryptology ePrint Archive 2023). https://eprint.iacr.org/2023/1364
- Dolev, Lynch, Pinter, Stark, Weihl (JACM 1986). https://doi.org/10.1145/5925.5931
- Fischer, Lynch, Paterson (JACM 1985). https://doi.org/10.1145/3149.214121
- Herlihy and Shavit (JACM 1999). https://doi.org/10.1145/331524.331529
- Ledent (PODC 2021). https://doi.org/10.1145/3465084.3467946
- Liu (OPODIS 2022). https://doi.org/10.4230/LIPIcs.OPODIS.2022.22
- Mendes, Herlihy, Vaidya, Garg (Distrib. Comput. 2015). https://doi.org/10.1007/s00446-014-0240-5
- Nowak and Rybicki (DISC 2019). https://doi.org/10.4230/LIPIcs.DISC.2019.29
- Saks and Zaharoglou (SICOMP 2000). https://doi.org/10.1137/S0097539796307698
- Schenk (FOCS 1995). https://doi.org/10.1109/SFCS.1995.492673