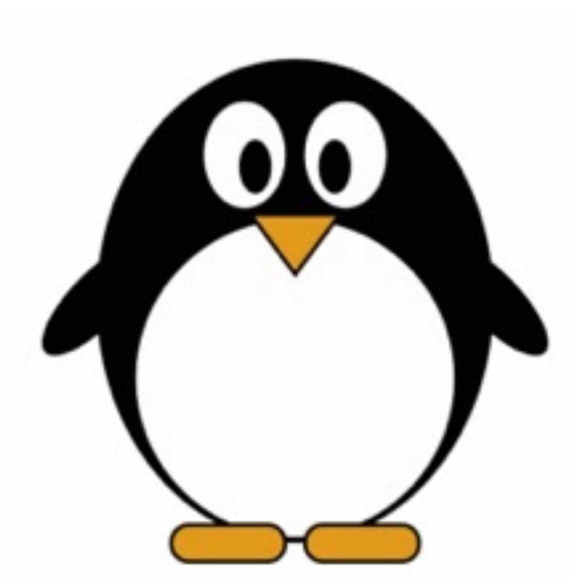
Computability in Population Protocols

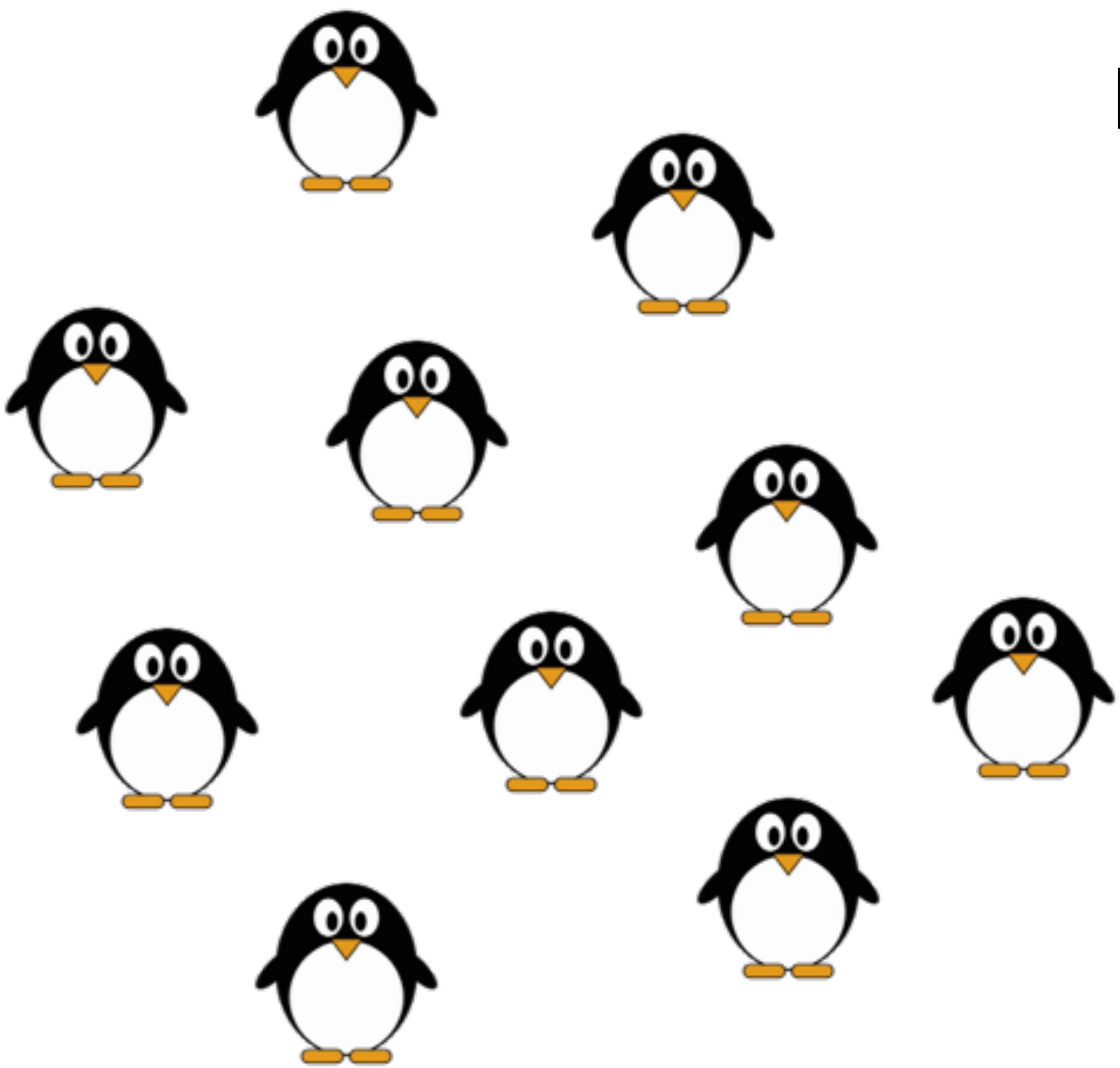
Peva Blanchard
EPFL 2014/2015

Population Protocol



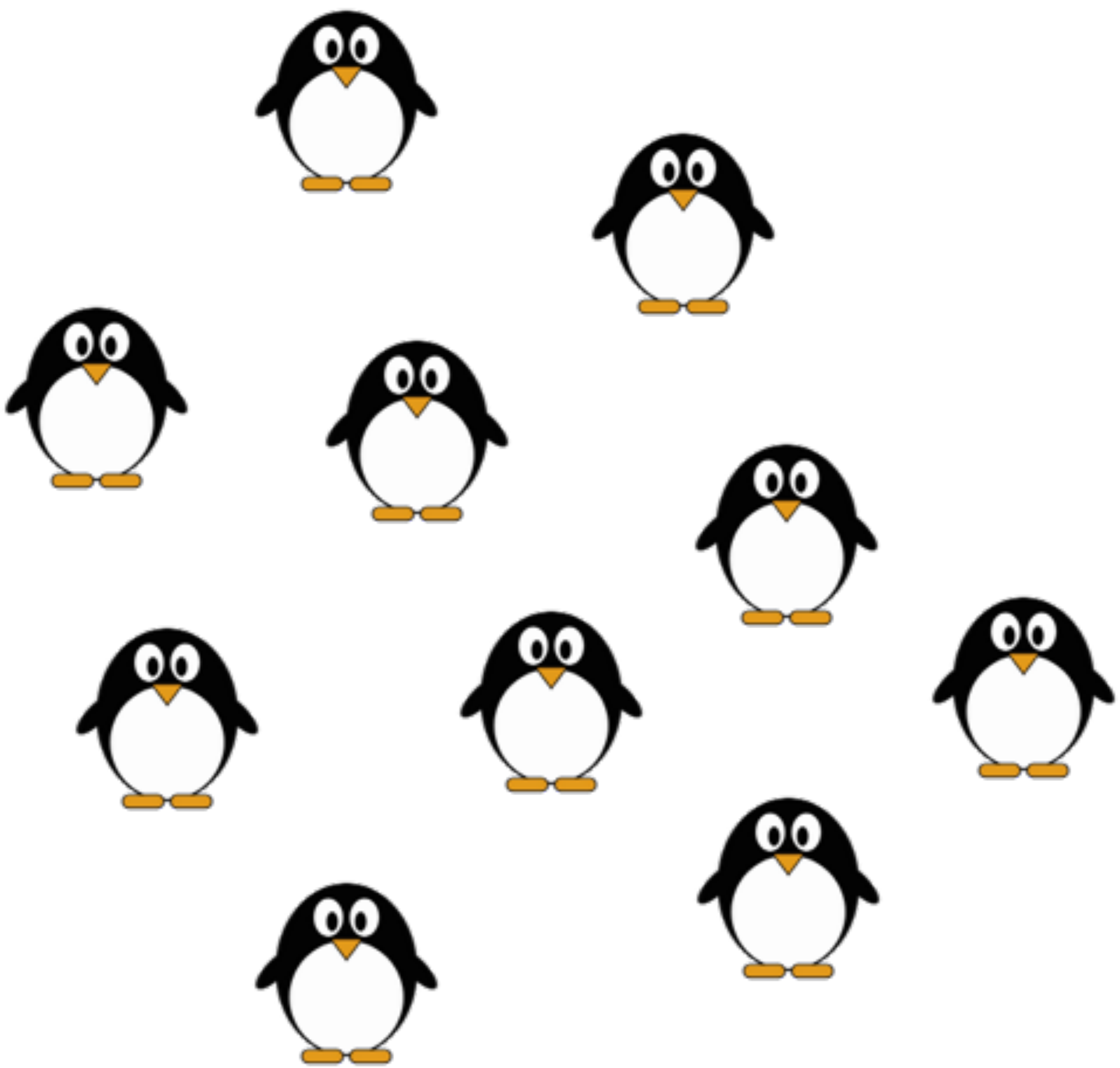
Agent : no id, small memory

Population Protocol



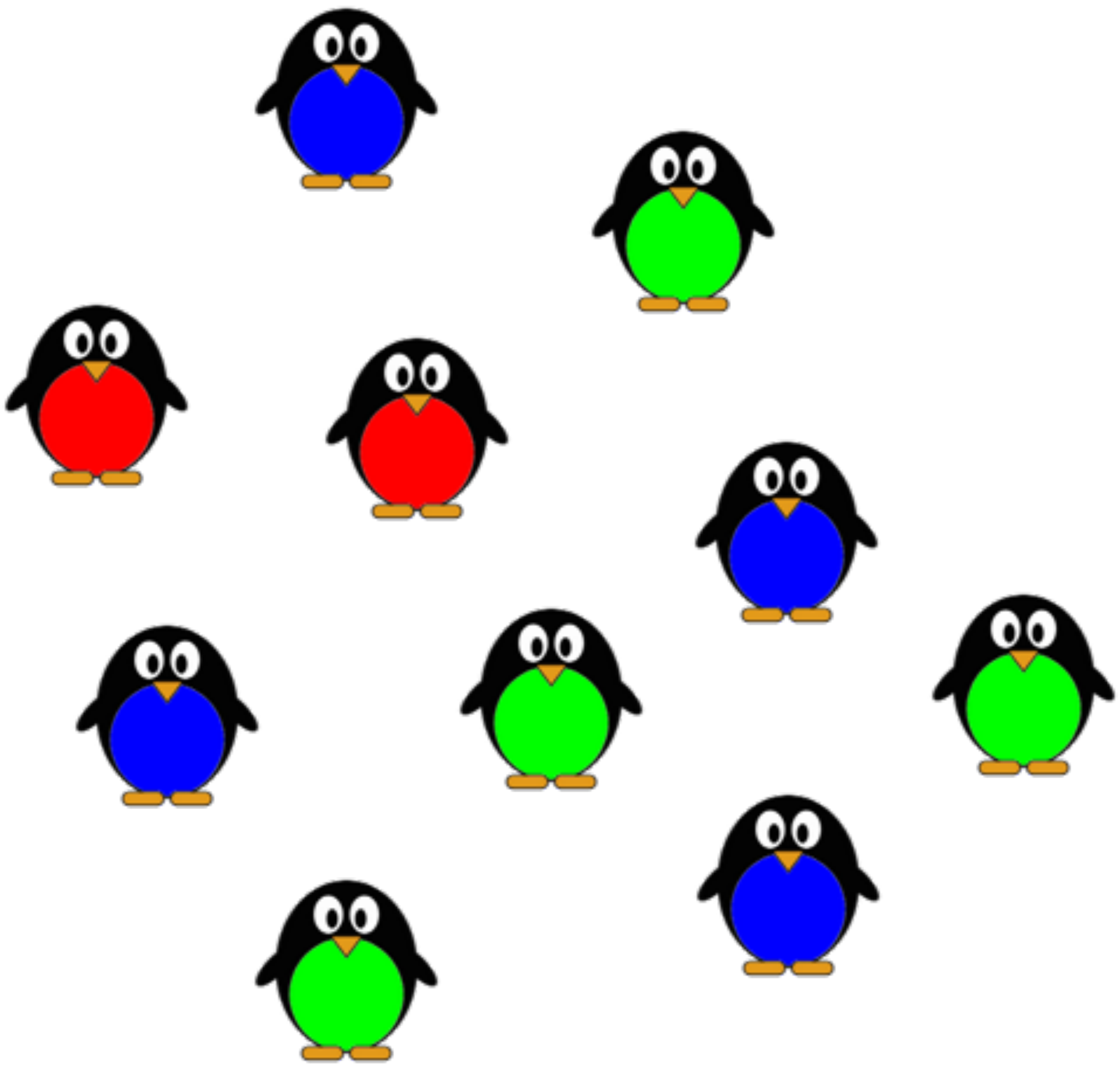
Population arbitrary size n

Population Protocol



Initially: sensors give data

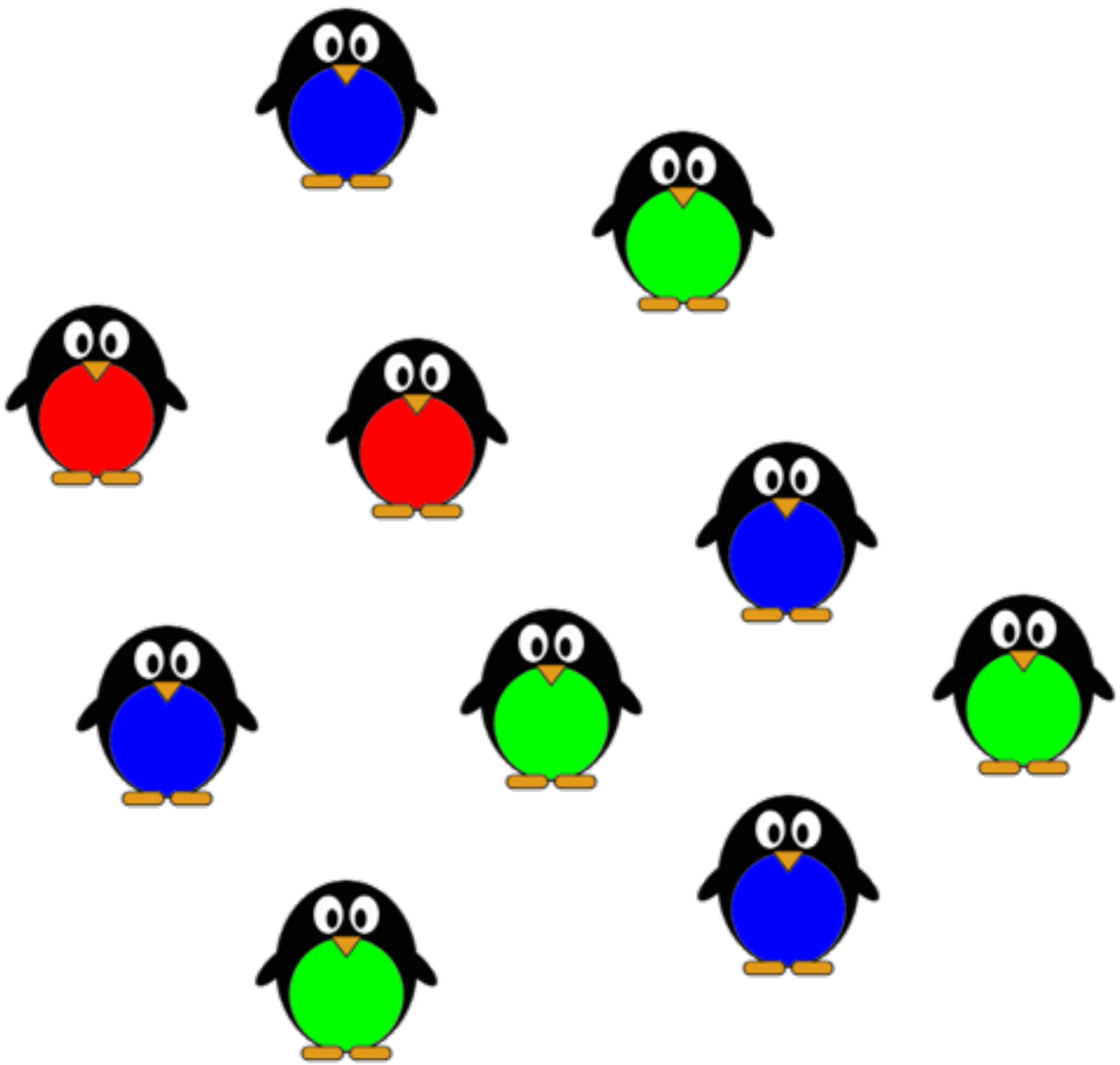
Population Protocol



Initially: sensors give data

red green blue

Population Protocol



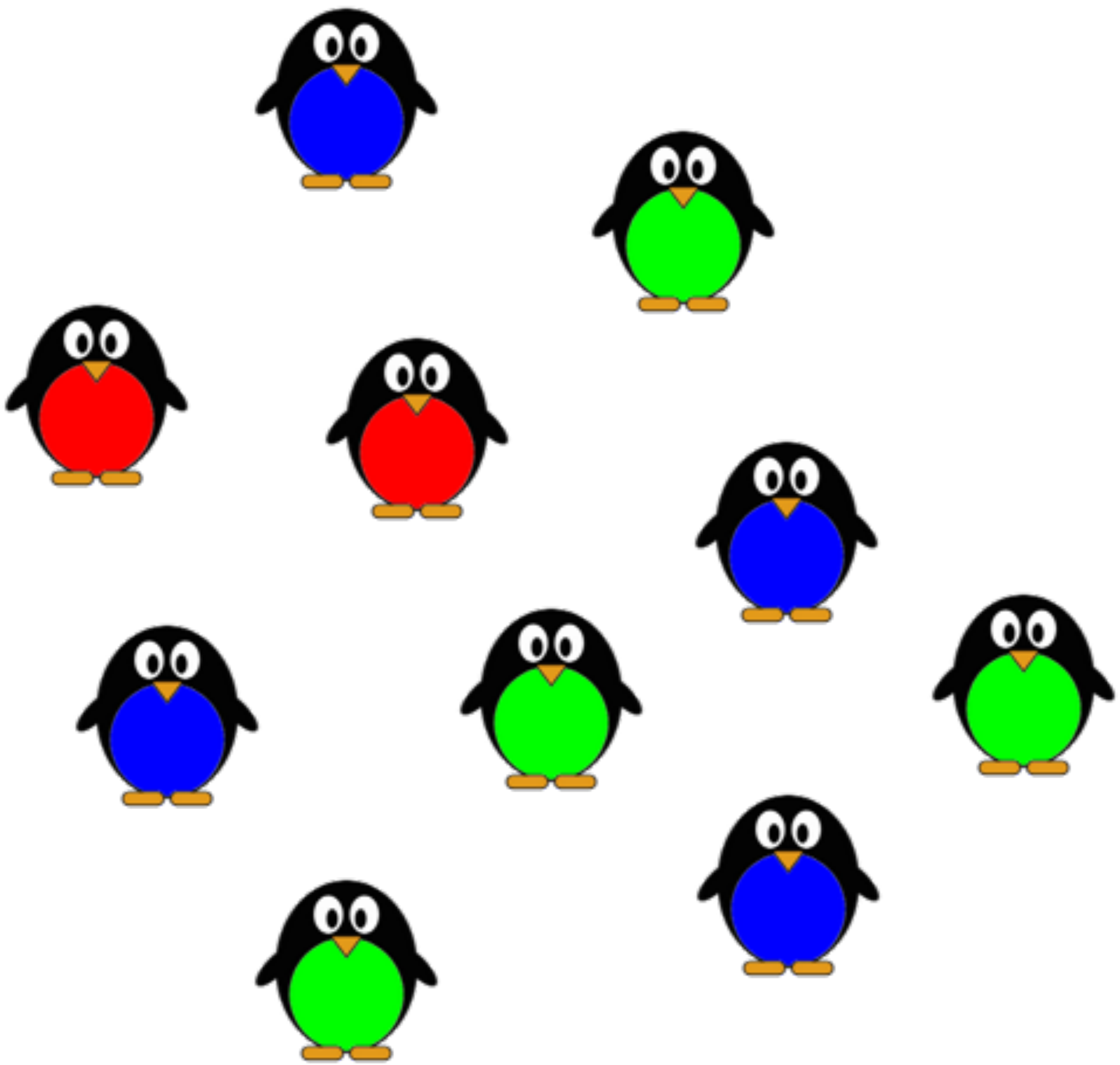
Initially: sensors give data

red green blue

↓

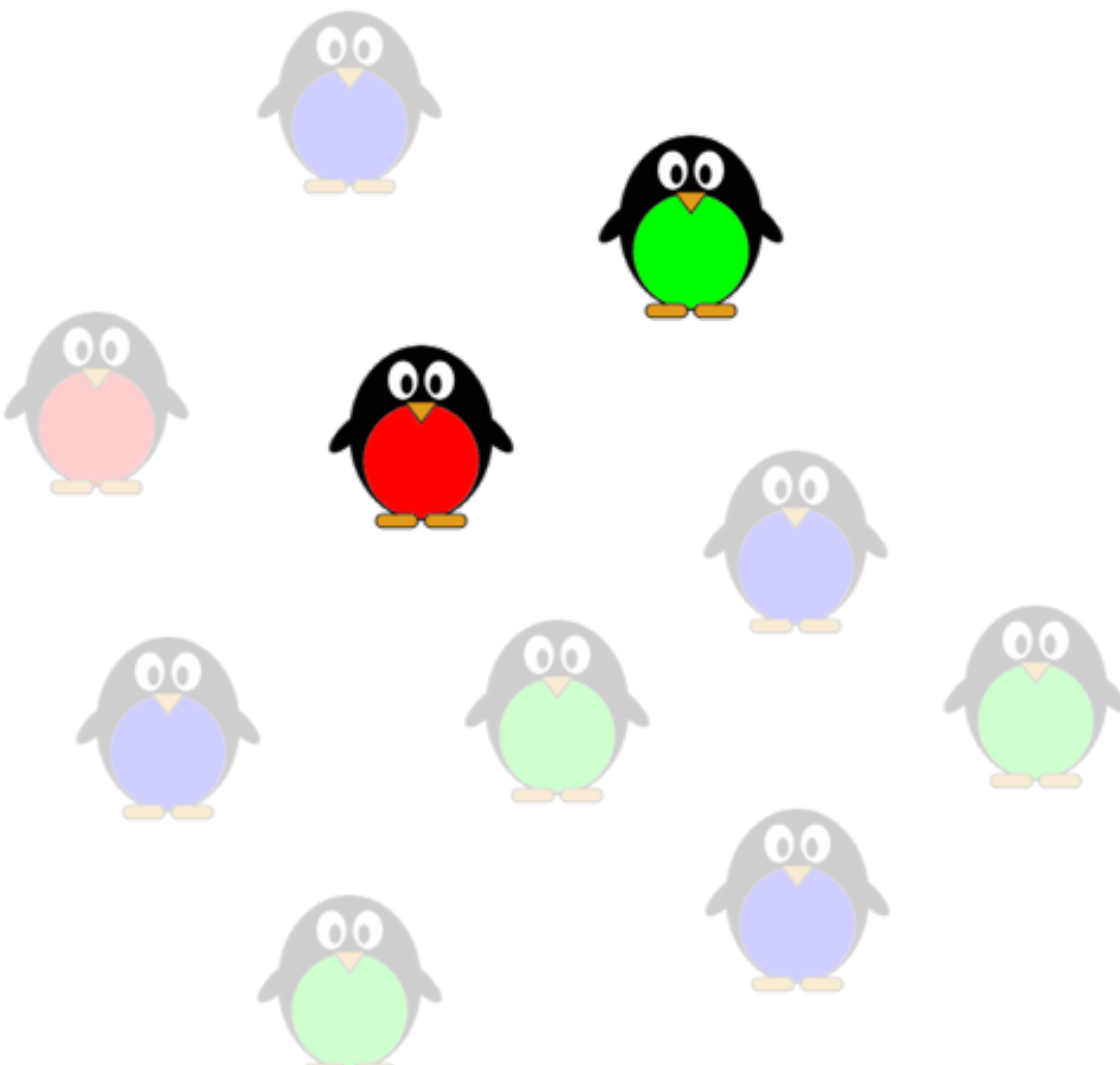
initial states

Population Protocol



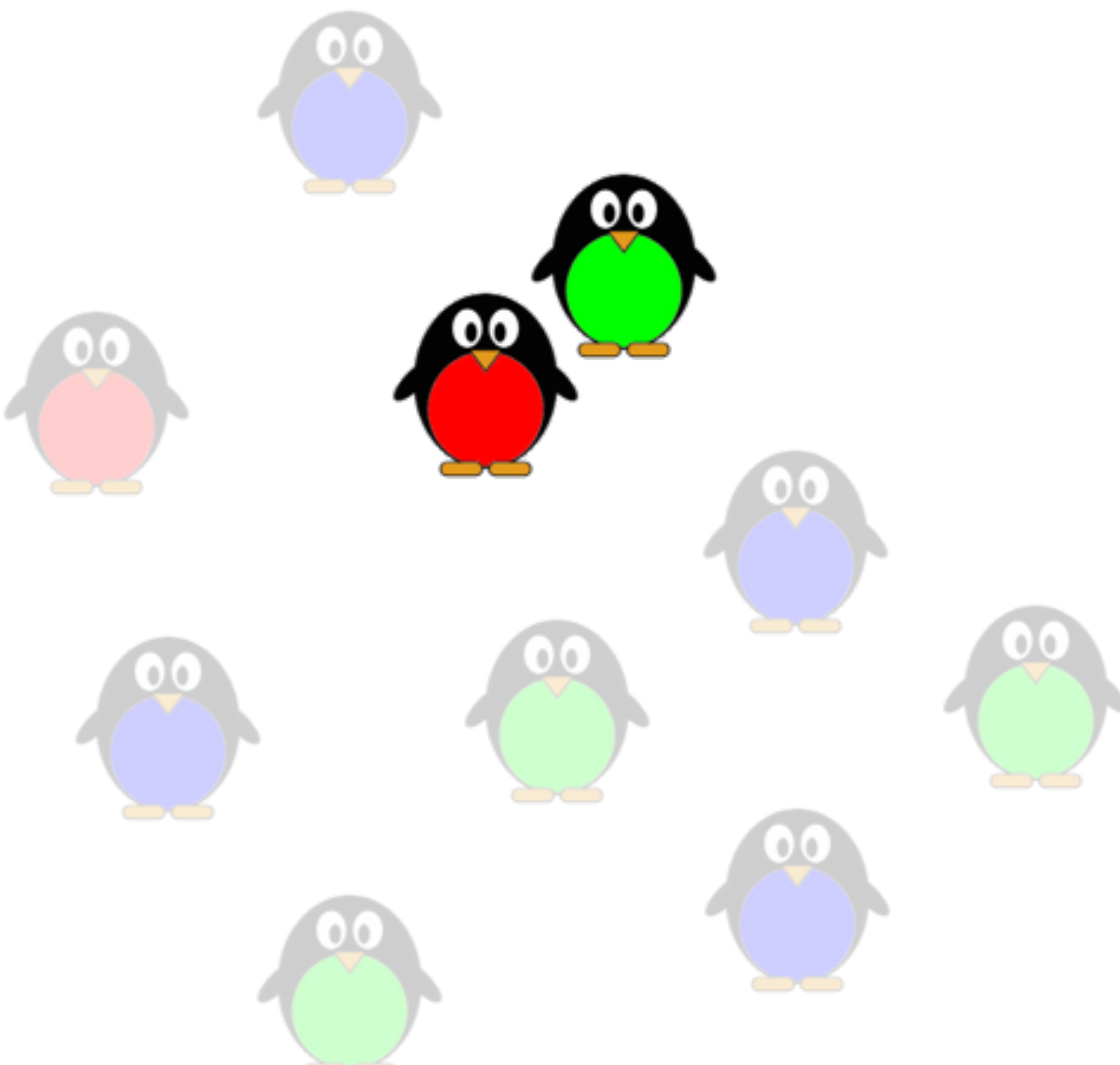
Agents move

Population Protocol



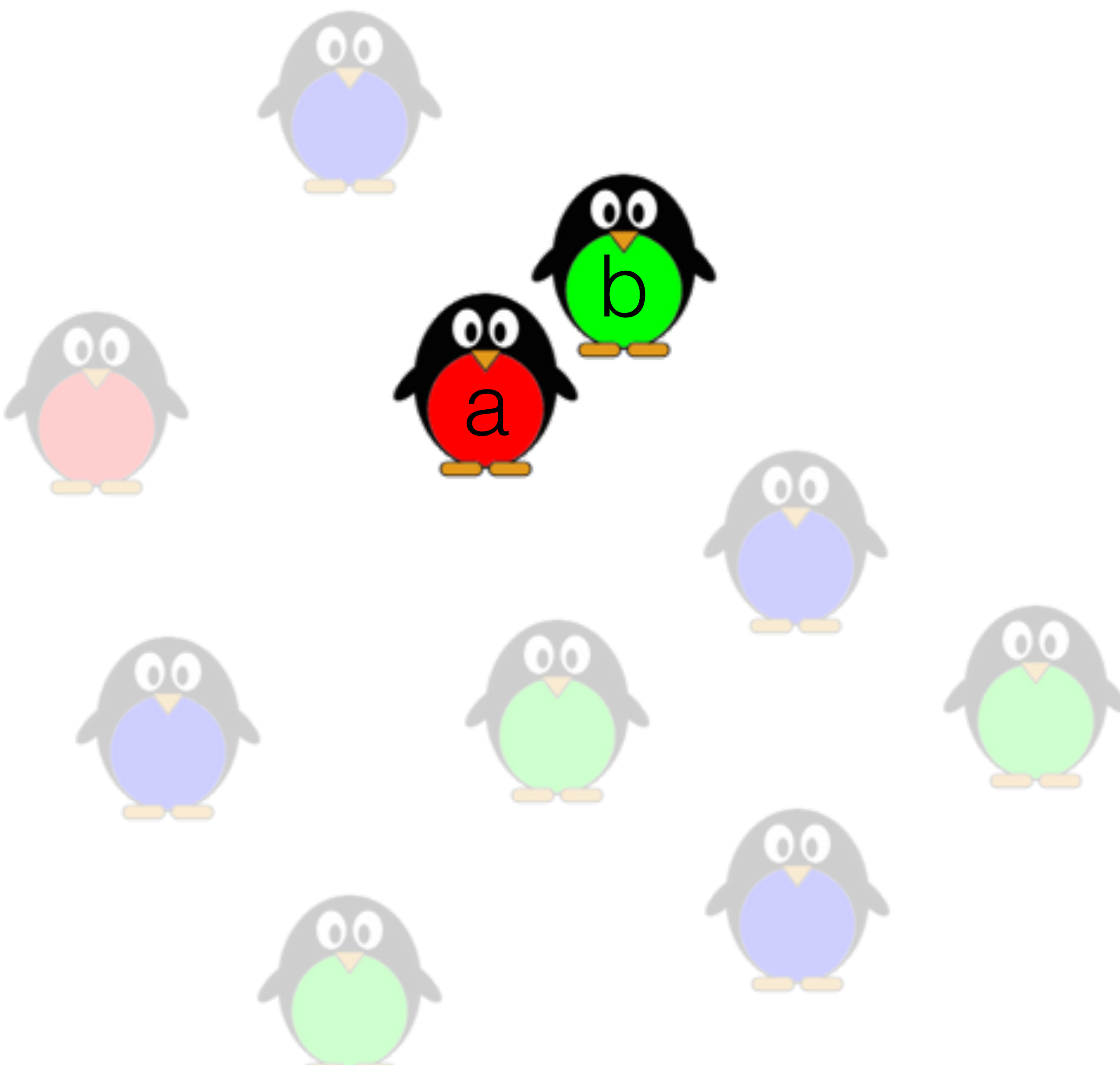
Agents move

Population Protocol



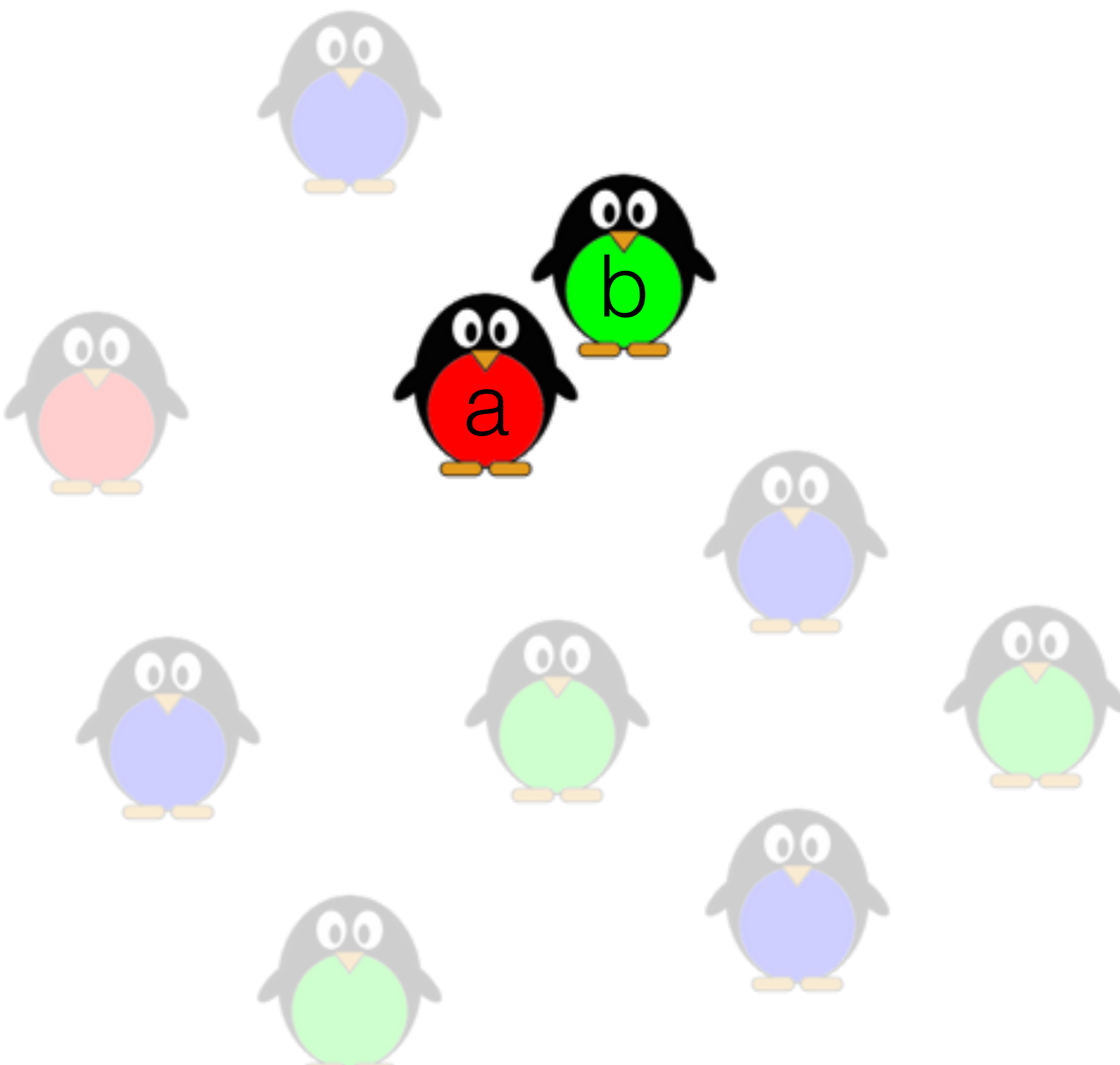
Agents move

Population Protocol



Agents move

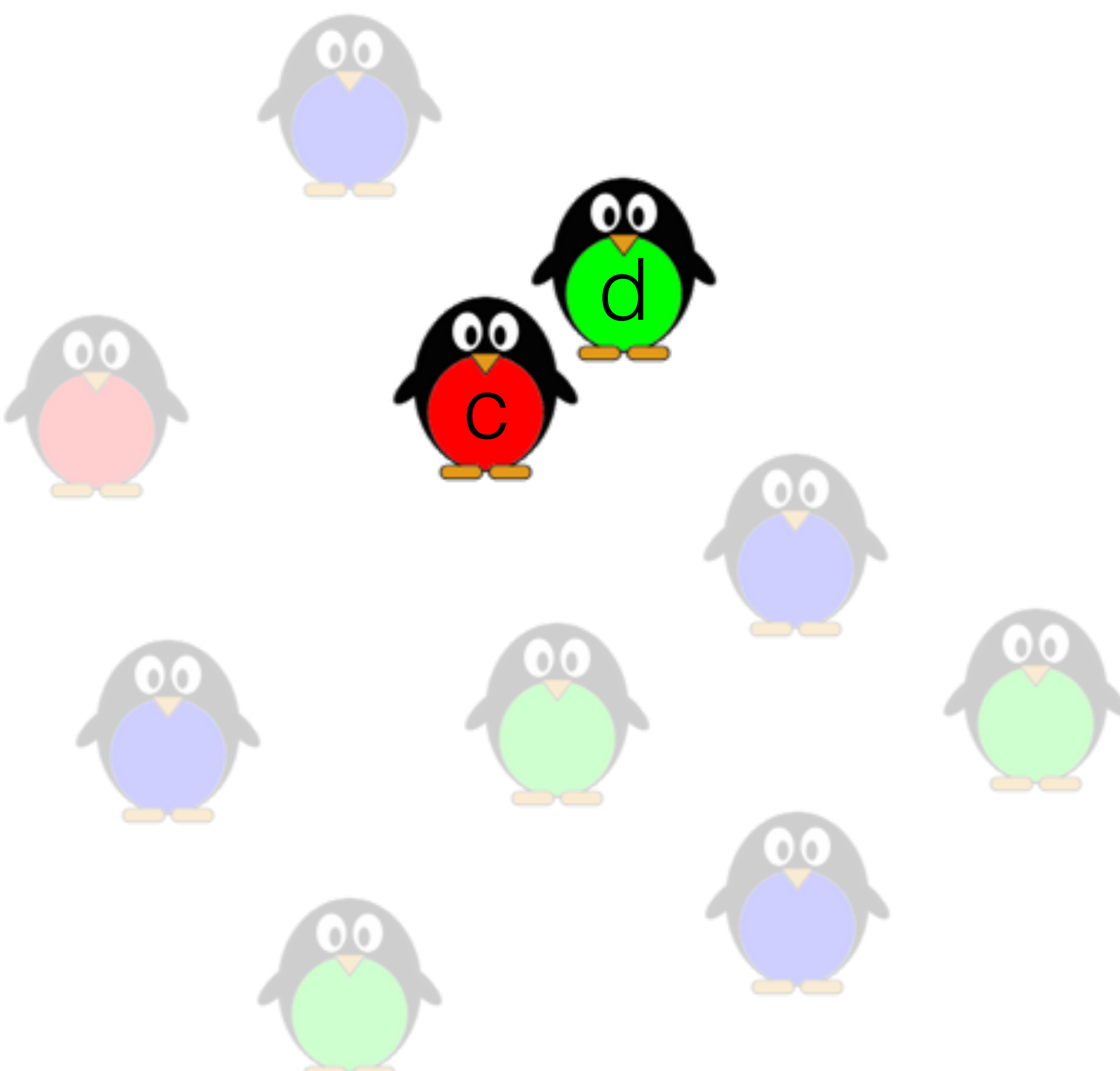
Population Protocol



Protocol rule

$a,b \longrightarrow c,d$

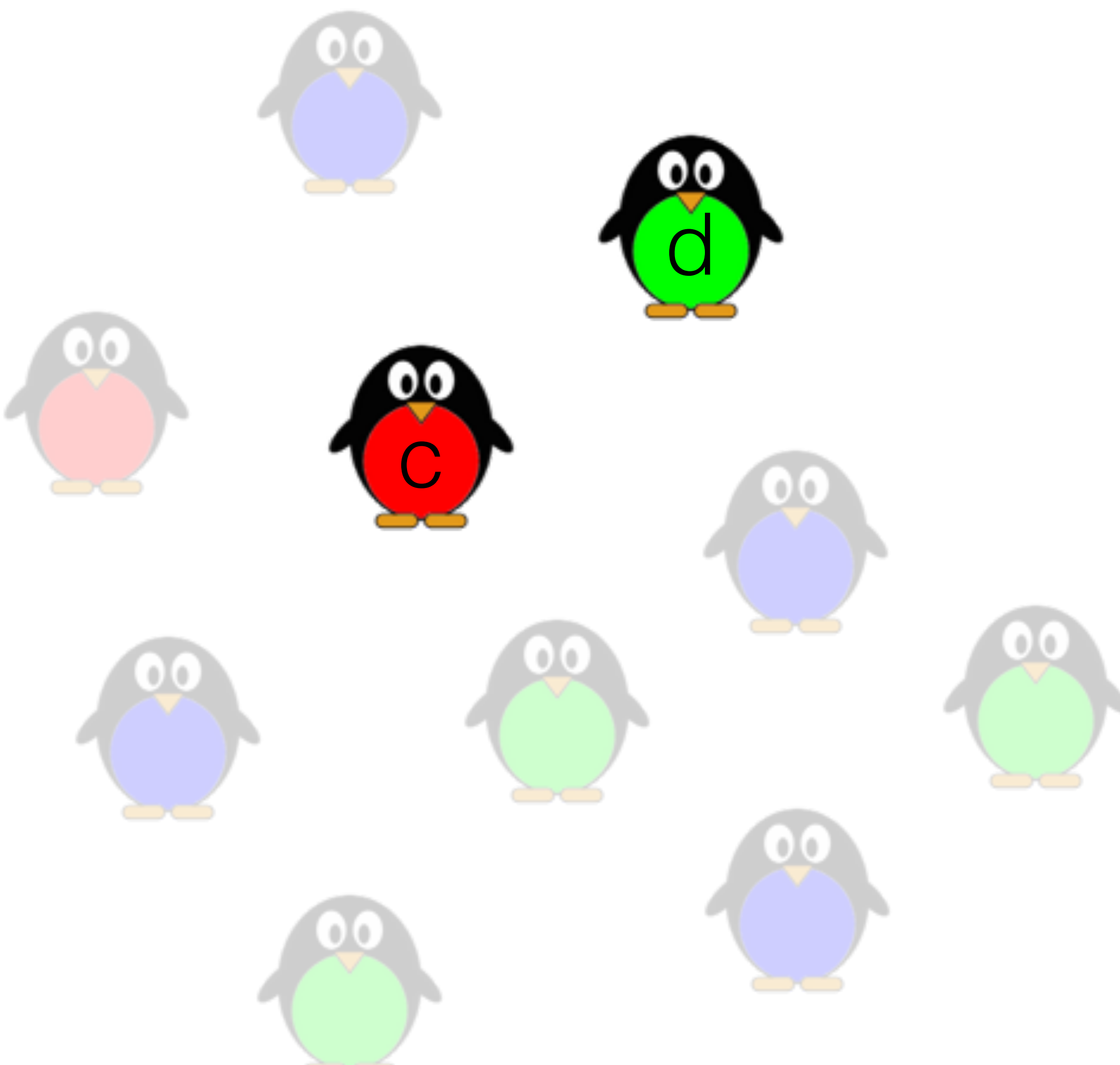
Population Protocol



Protocol rule

$$a,b \longrightarrow c,d$$

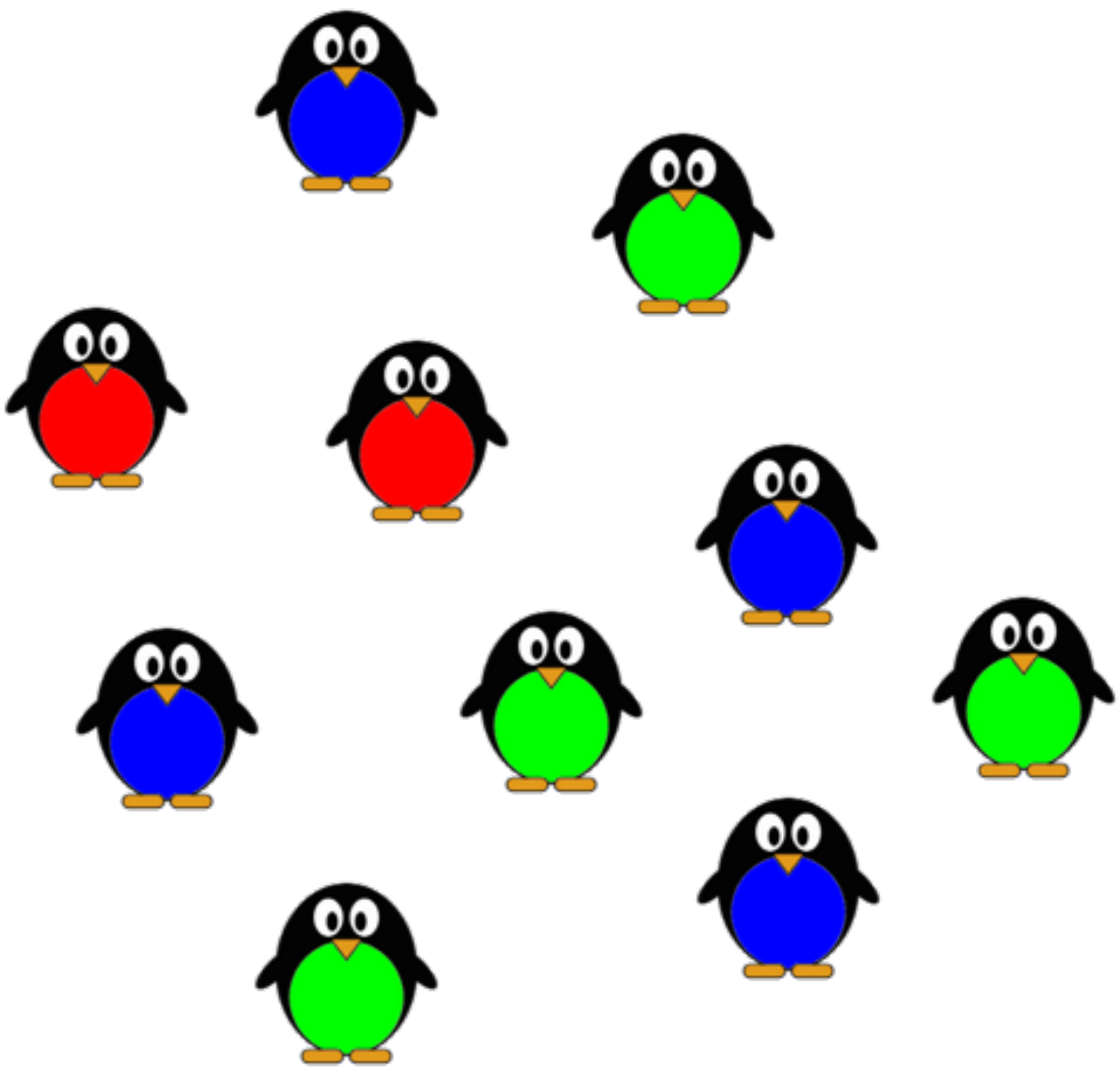
Population Protocol



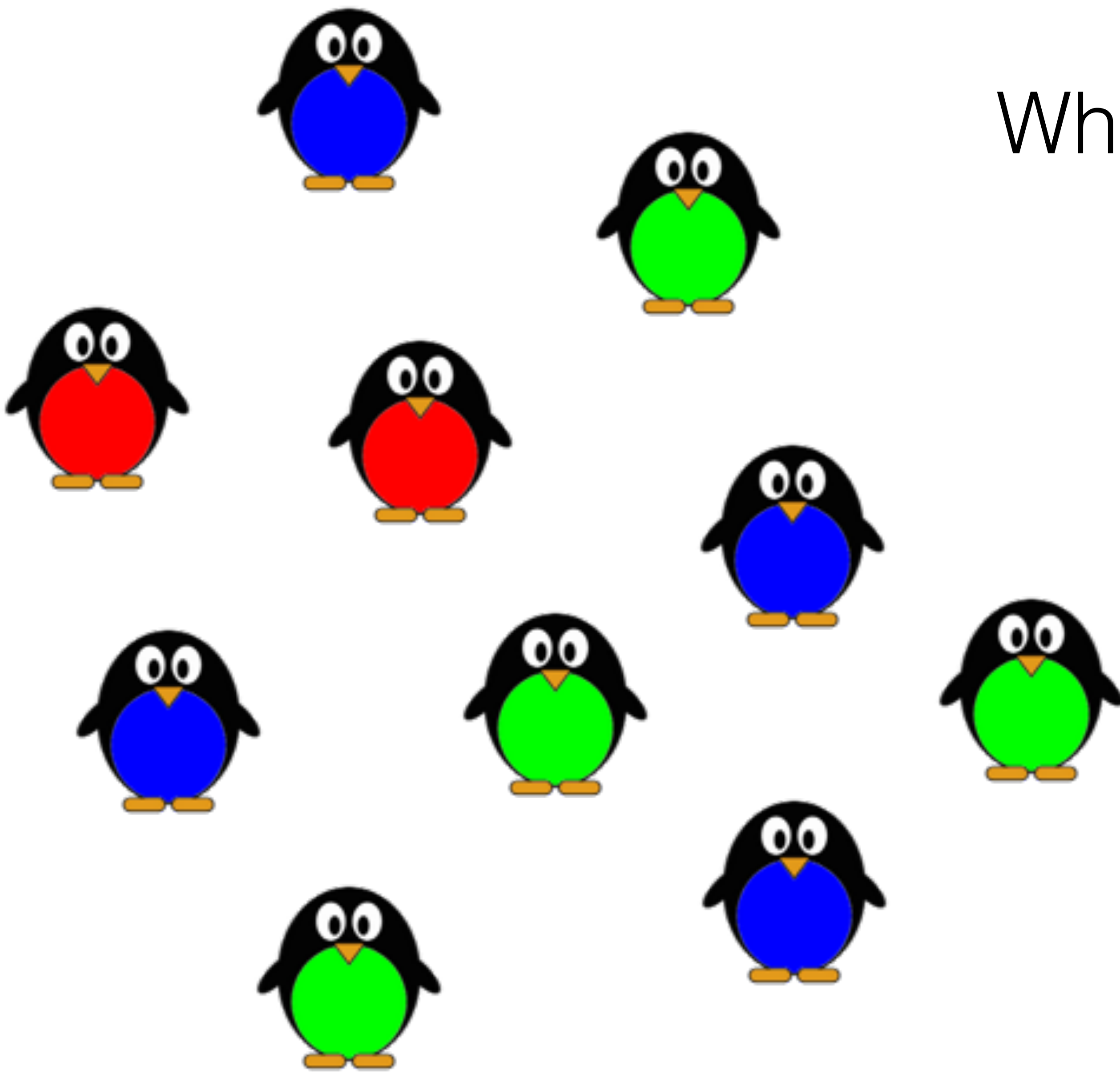
Protocol rule

a,b ⟶ c,d

What is computable ?

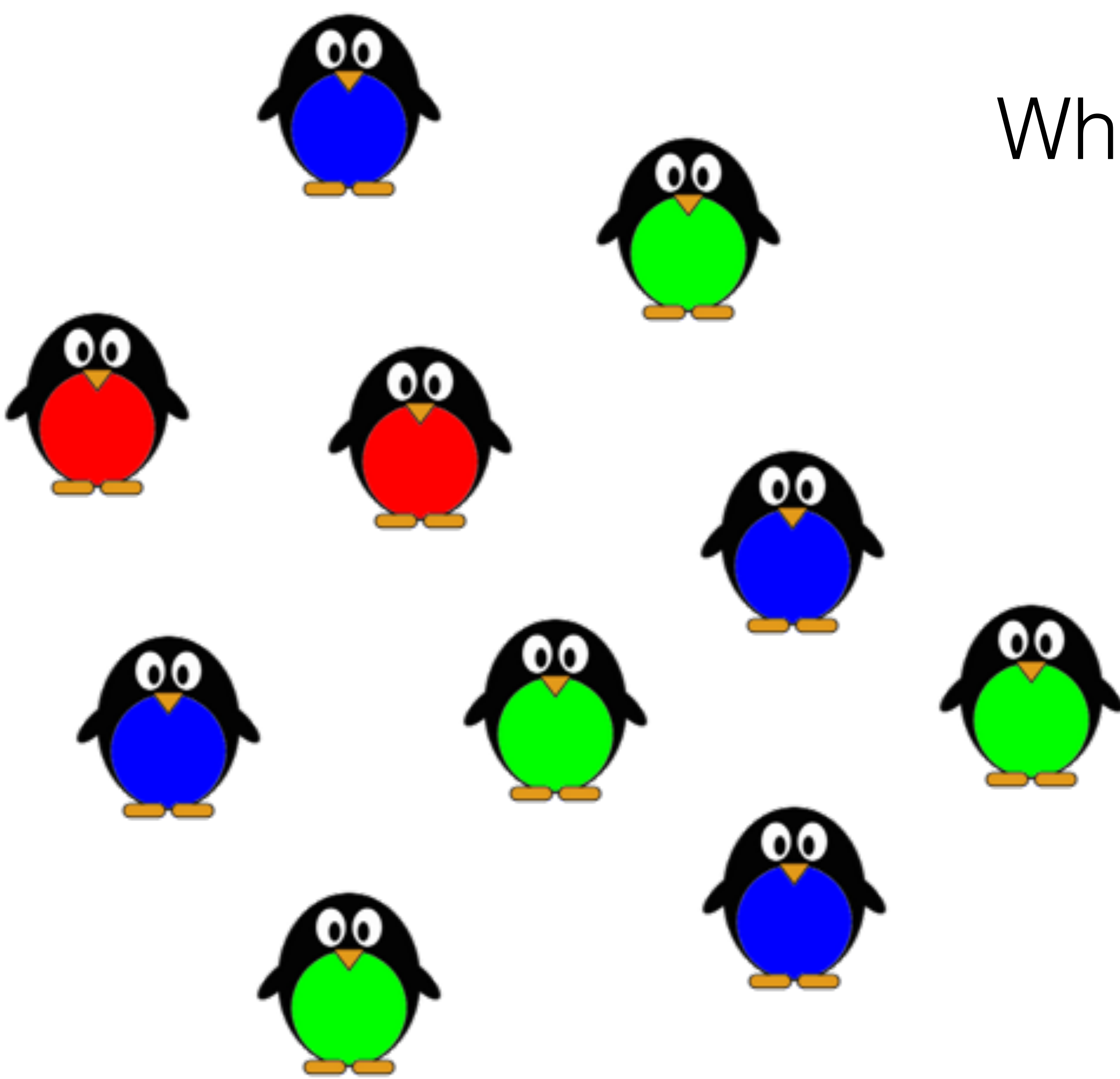


What is computable ?



What can the agents know about the initial configuration ?

What is computable ?



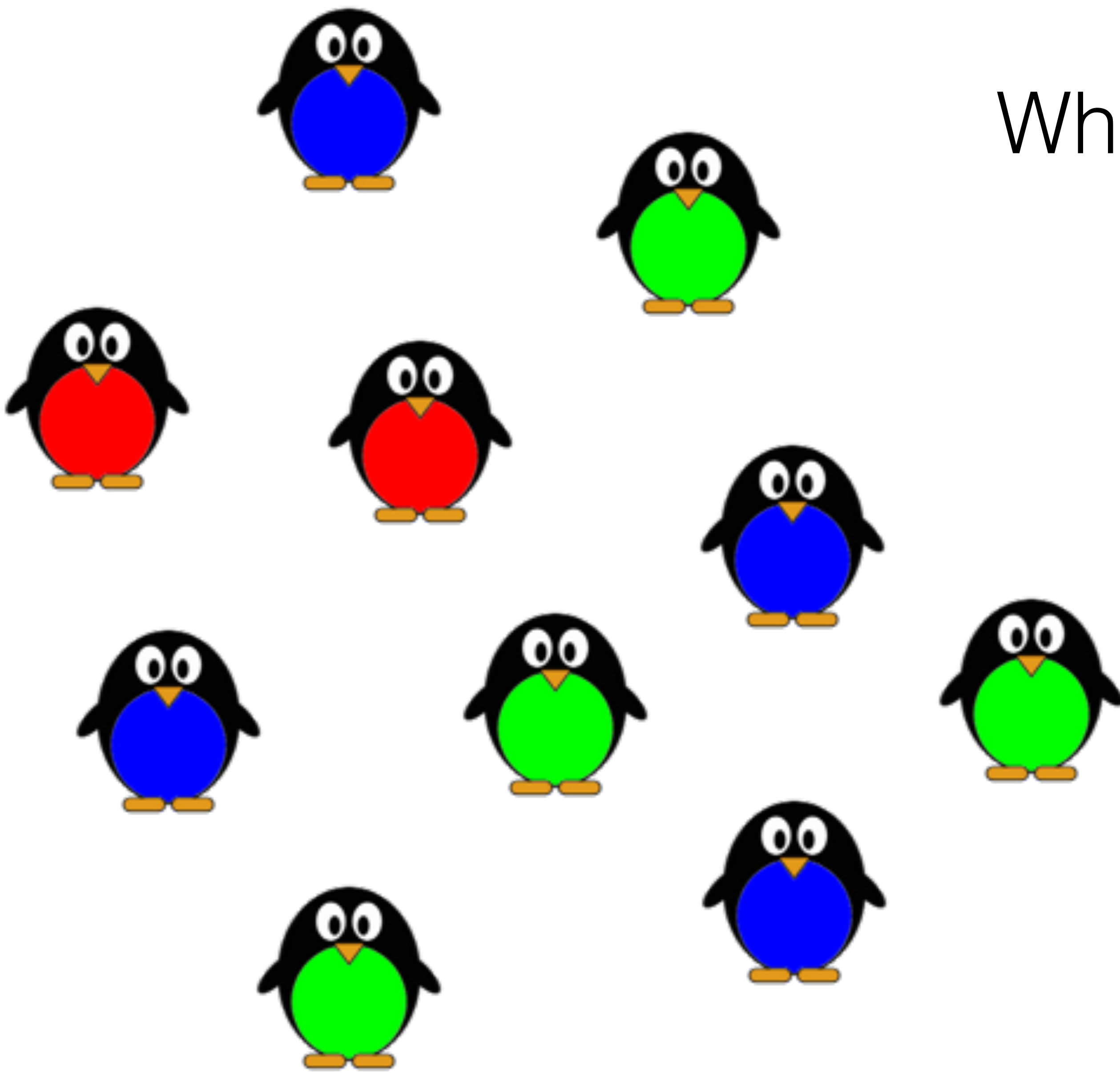
What can the agents know about the initial configuration ?

No id

$\Downarrow$

Only numbers of "species"

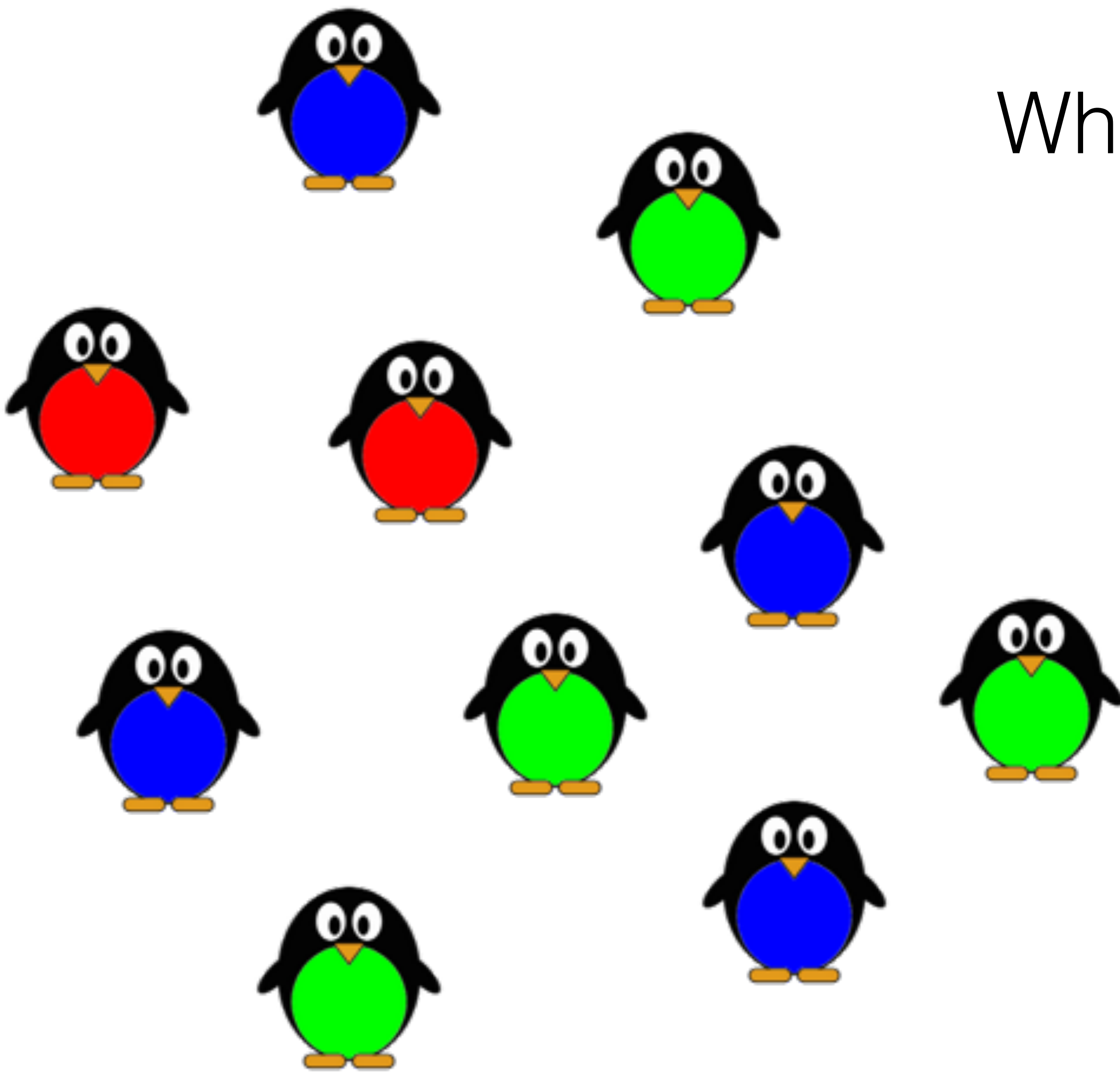
What is computable ?



What can the agents know about the initial configuration ?

#green $\leq$ 4

What is computable ?

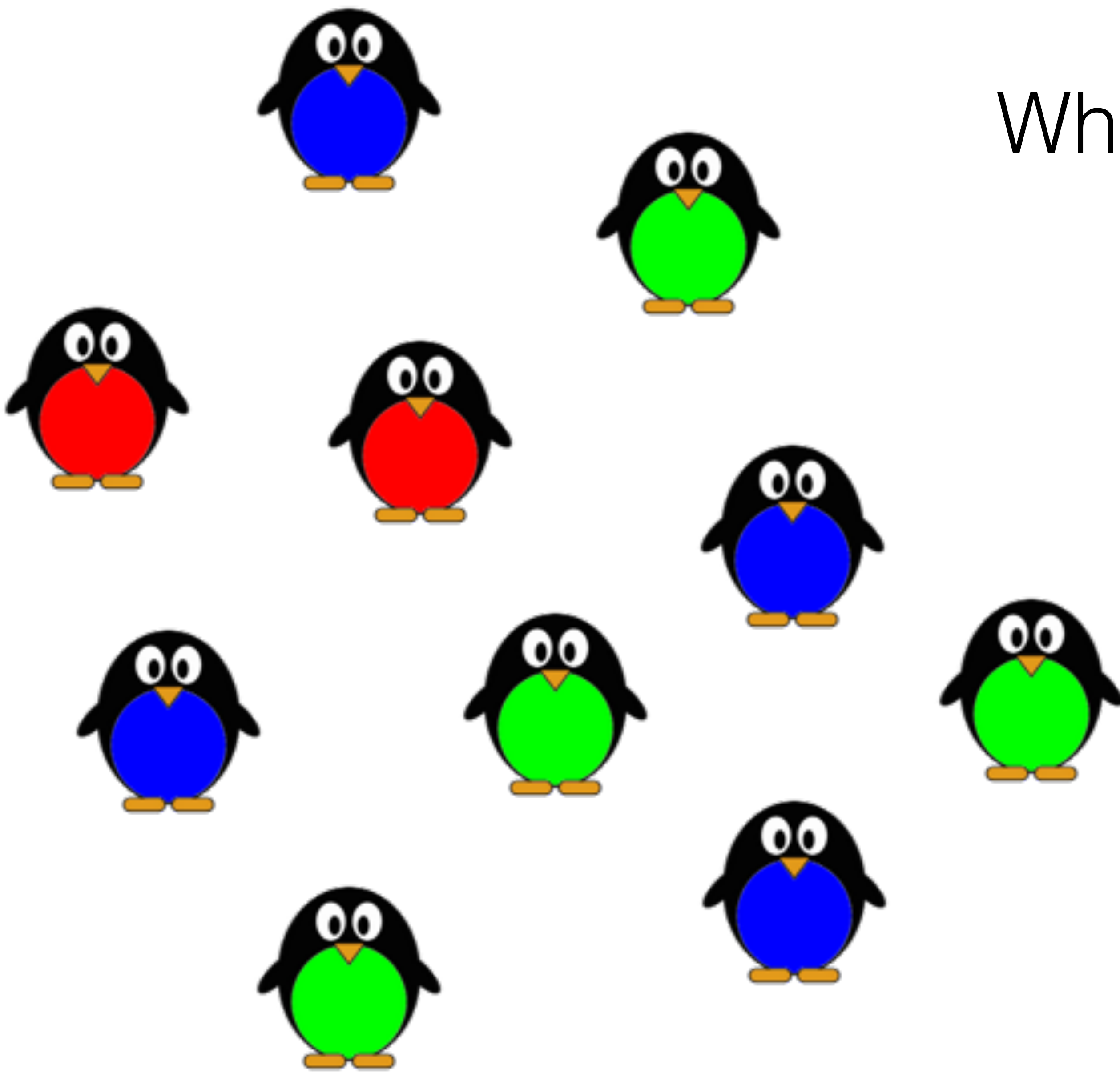


What can the agents know about the initial configuration ?

$\#green \leq 4$

$\#green \leq \#blue$

What is computable ?



What can the agents know about the initial configuration ?

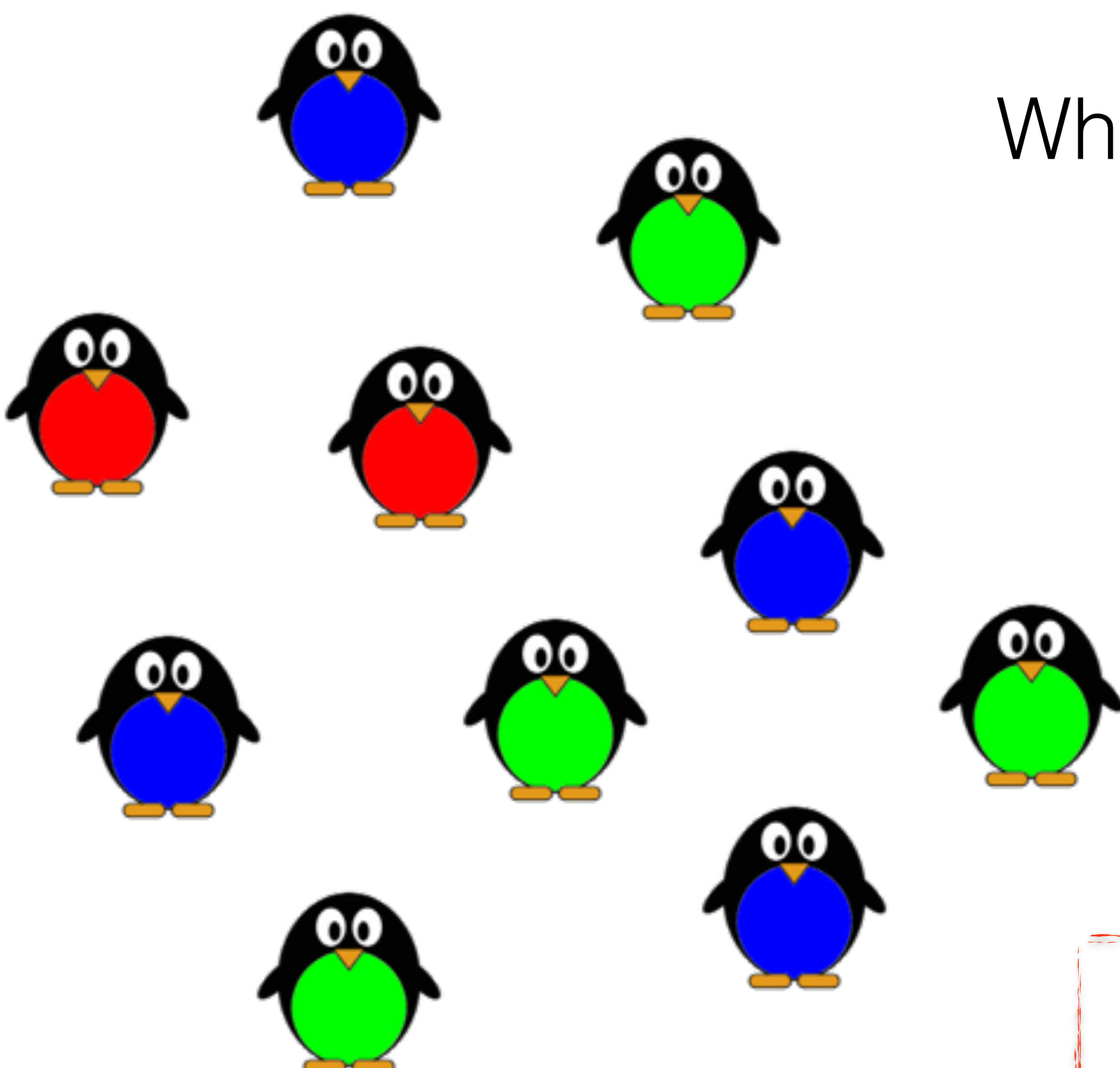
$\#green \leq 4$

$\#green \leq \#blue$

$\#green \leq \#blue + 2\#red$

$\#green \leq \#blue \times \#red$

What is computable ?



What can the agents know about the initial configuration ?

$\#green \leq 4$

$\#green \leq \#blue$

$\#green \leq \#blue + 2\#red$

OK

$\#green \leq \#blue \times \#red$

nOK

predicate P computable

predicate P computable

There exists a protocol A such that

predicate P computable

There exists a protocol A such that

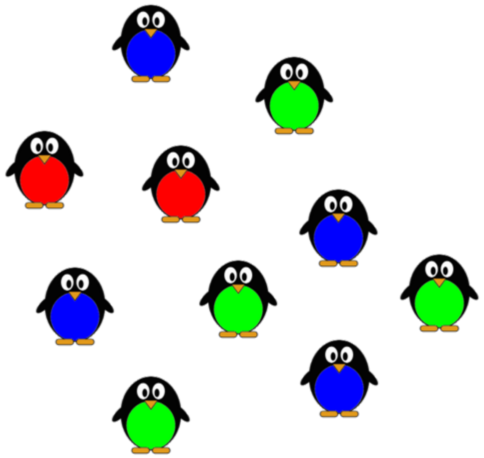
for any population size

predicate P computable

There exists a protocol A such that

for any population size

for any input assignment

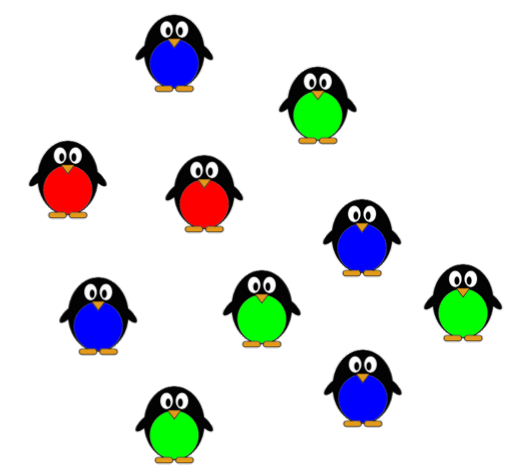


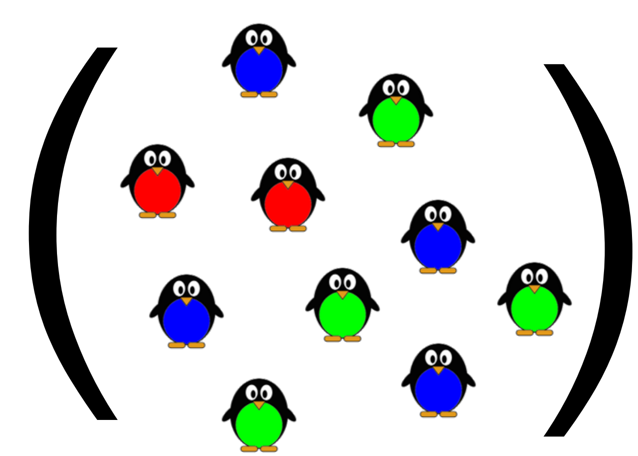
predicate P computable

There exists a protocol A such that

for any population size

for any input assignment



eventually **all** agents output P 

$$\#\text{green} - 2\ \#\text{blue} \leq 4 \qquad \text{(naive)}$$

$\#green - 2\,\#blue \leq 4$ (naive)

Assume a unique leader

with counter, initially 0

#green - 2 #blue $\leq$ 4 (naive)

Assume a unique leader

with counter, initially 0

,

→ counter -= 2

$\#\text{green} - 2\,\#\text{blue} \leq 4$ (naive)

Assume a unique leader

with counter, initially 0

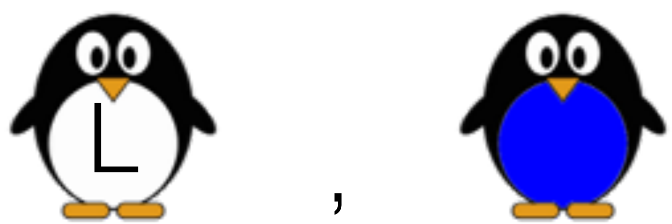
,  → counter -= 2

,  → counter += 1

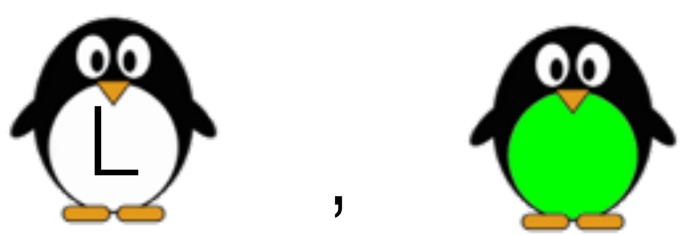
#green - 2 #blue $\leq$ 4 (naive)

Assume a unique leader

with counter, initially 0



→ counter -= 2



→ counter += 1

and mark them as seen.

$\#green - 2\,\#blue \leq 4$ (naive)

Assume a unique leader

with counter, initially 0

,  ⟶ counter -= 2

,  ⟶ counter += 1

and mark them as seen.

Leader output 1 iff counter $\leq$ 4

other agents copy leader's output.

#green - 2 #blue $\leq$ 4 (naive)

Assume a unique leader

with counter, initially 0

#1. How to elect a leader ?

BUT

,

→ counter -= 2

,

→ counter += 1

#2. How to bound memory ?

and mark them as seen.

Leader output 1 iff counter $\leq$ 4

other agents copy leader's output.

$\#green - 2\,\#blue \leq 4$

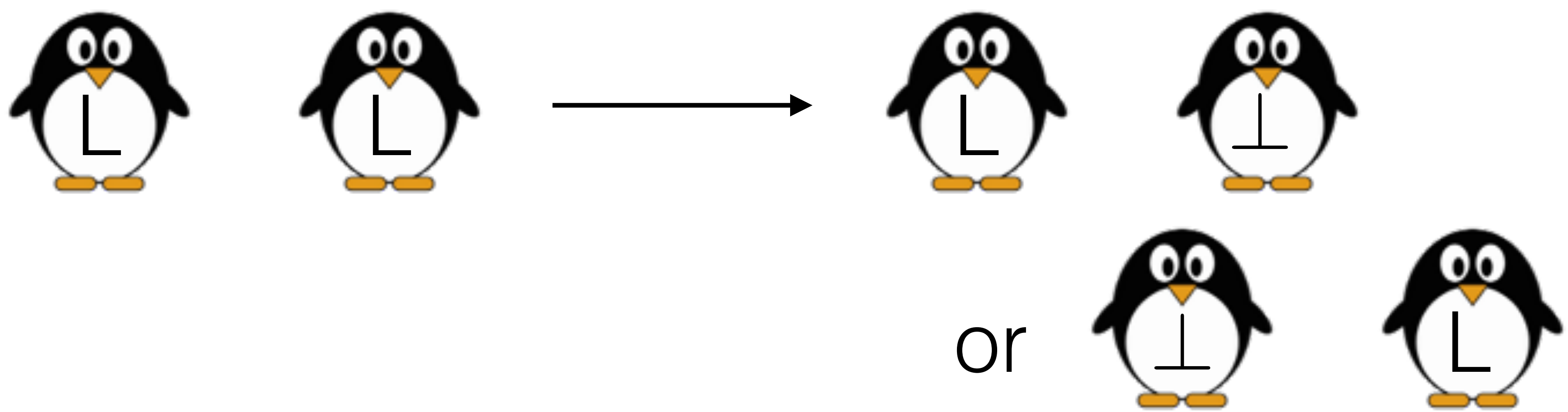
Leader election: each agent has a leader bit

initially, all leaders

$\#green - 2\,\#blue \leq 4$

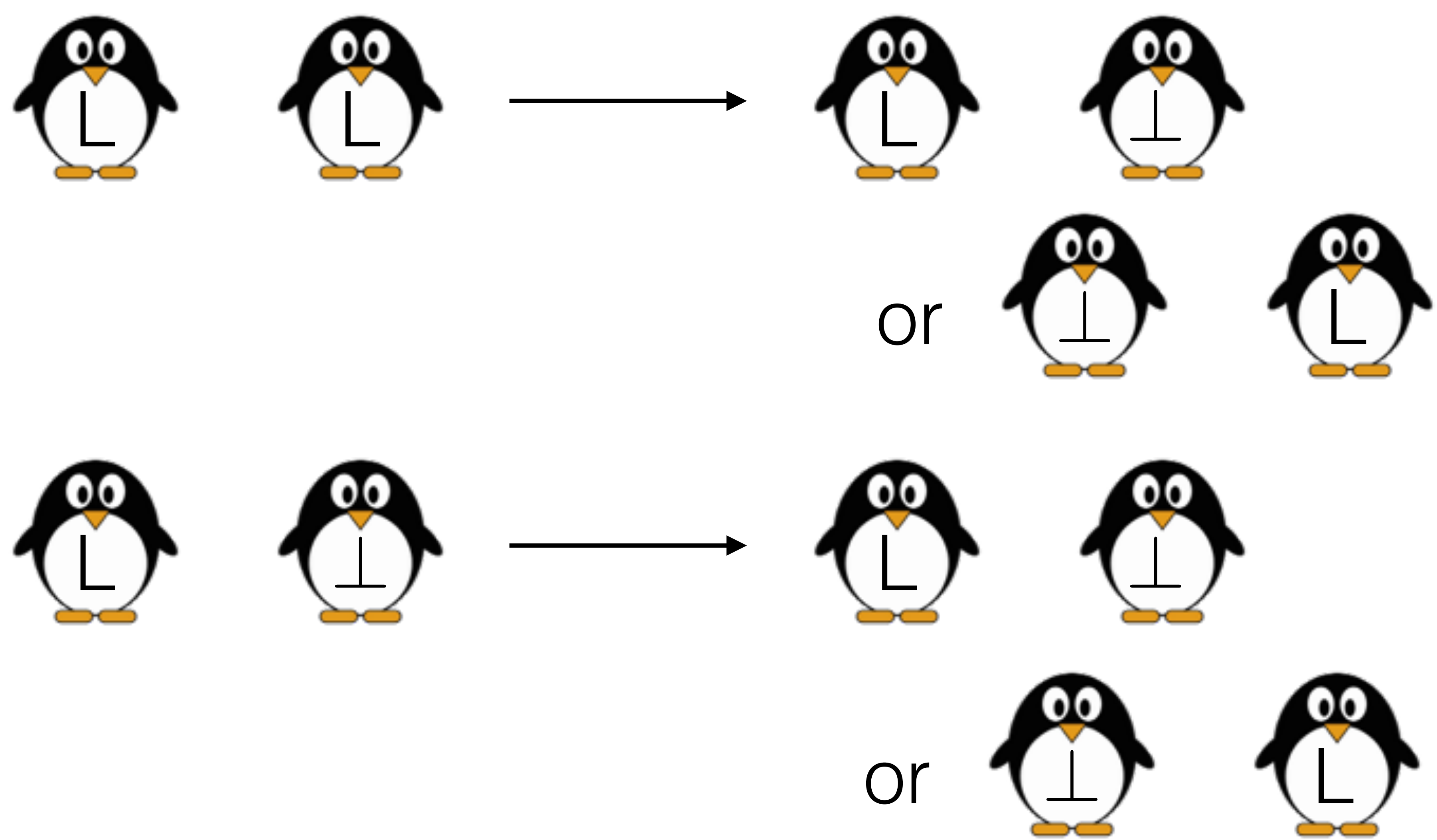
Leader election: each agent has a leader bit

initially, all leaders



#green - 2 #blue $\leq$ 4

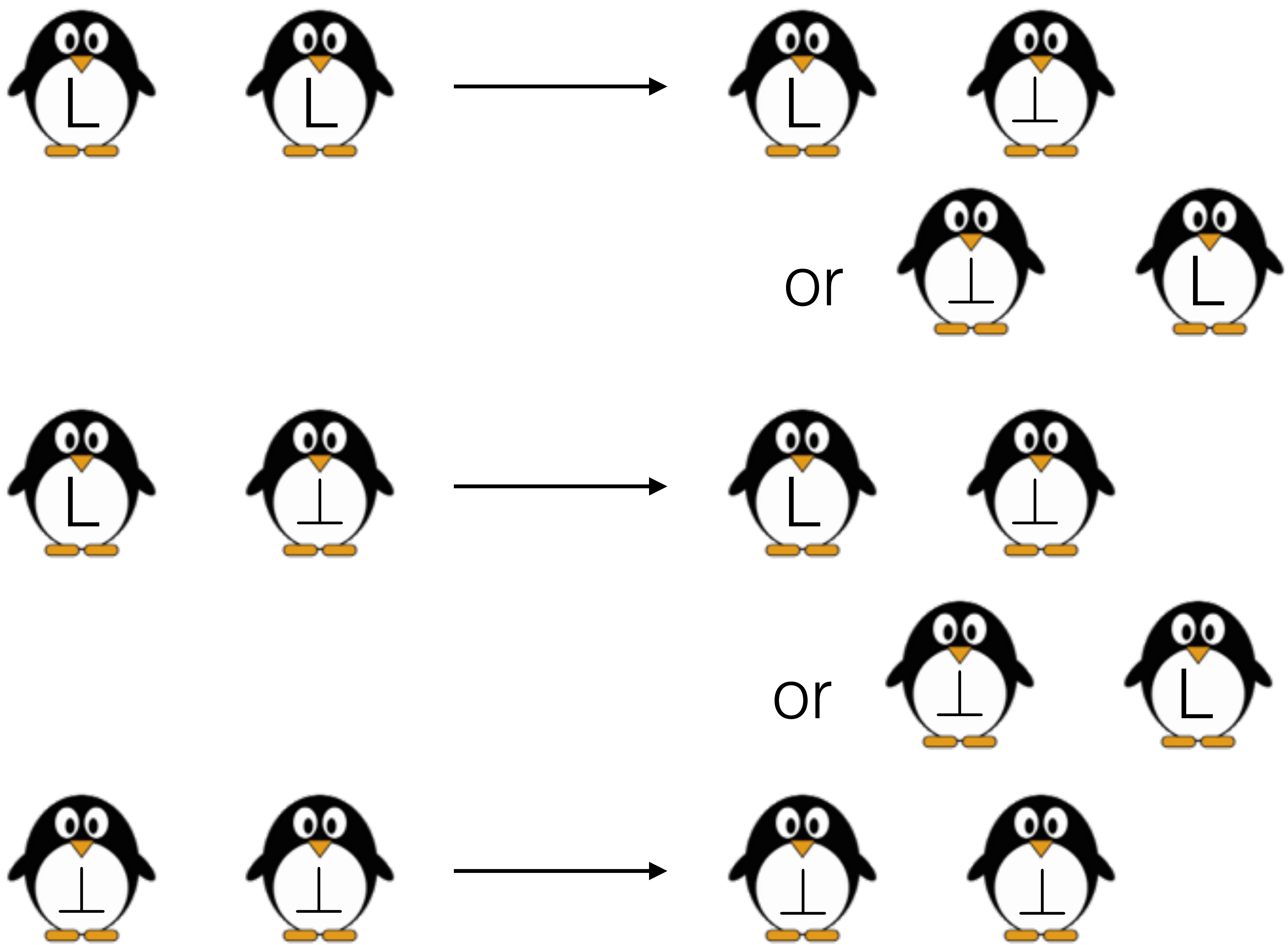
Leader election: each agent has a leader bit
initially, all leaders



$\#green - 2\ \#blue \leq 4$

Leader election: each agent has a leader bit

initially, all leaders



#green - 2 #blue $\leq$ 4

Counter issue

Fix a large enough limit, e.g. $s \geq 5$

#green - 2 #blue $\leq$ 4

Counter issue

Fix a large enough limit, e.g. $s \geq 5$

All agents have counter $-s \leq u \leq s$

#green - 2 #blue $\leq$ 4

Counter issue

Fix a large enough limit, e.g. $s \geq 5$

All agents have counter $-s \leq u \leq s$

initially

$u_{init} = -2$

$u_{init} = 1$

#green - 2 #blue $\leq$ 4

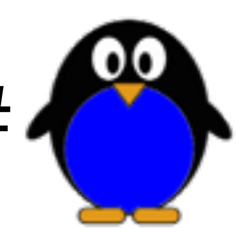
Counter issue

Fix a large enough limit, e.g. $s \geq 5$

All agents have counter $-s \leq u \leq s$

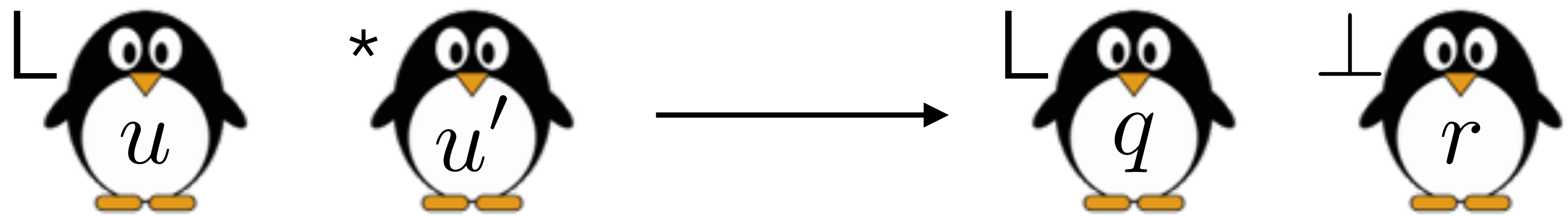
initially  $u_{init} = -2$

 $u_{init} = 1$

$\sum_{\text{agents}} u =$ #  - 2 # 

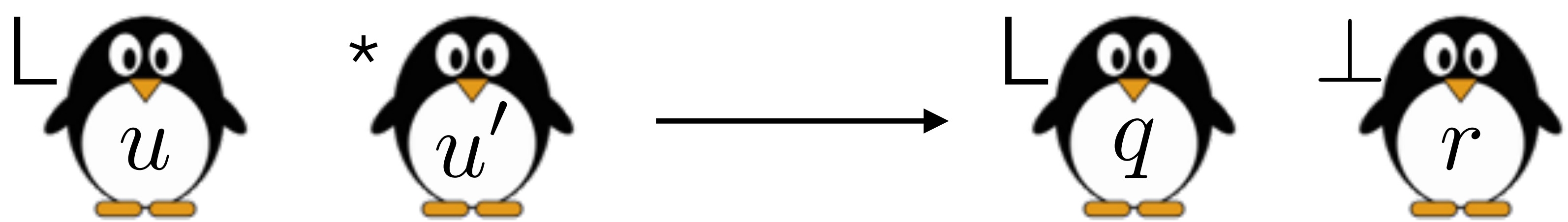
#green - 2 #blue $\leq$ 4

Counter issue



#green - 2 #blue $\leq$ 4

Counter issue

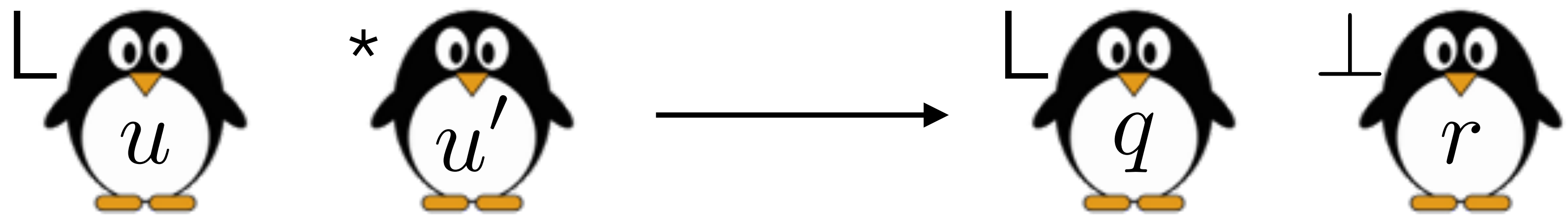


(naive)

$$q(u, u') = u + u'$$
$$r(u, u') = 0$$

#green - 2 #blue $\leq$ 4

Counter issue



truncated sum

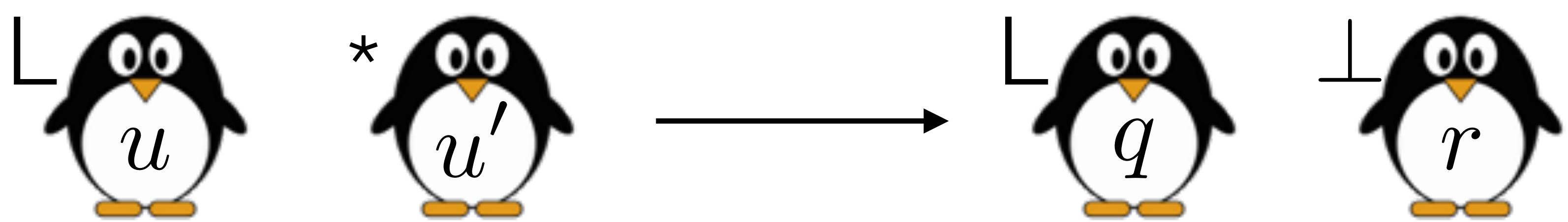
$$q(u,u') = \max\{-s, \min\{s, u+u'\}\}$$

$$r(u,u') = u + u' - q(u,u')$$

remainder

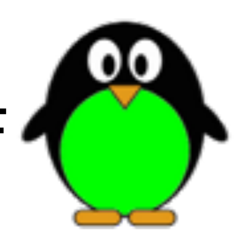
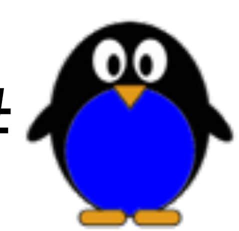
#green - 2 #blue $\leq$ 4

Counter issue



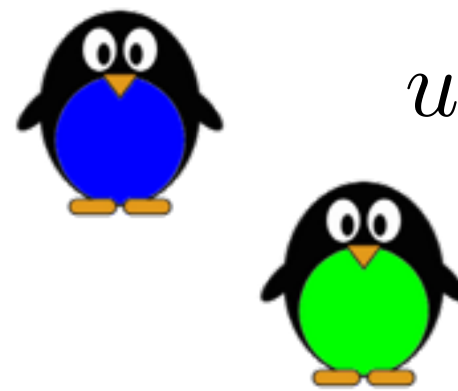
$$q(u,u') = \max\{-s, \min\{s, u+u'\}\}$$ truncated sum

$$r(u,u') = u + u' - q(u,u')$$ remainder

Invariant $\sum_{\text{agents}} u =$ #  - 2 # 

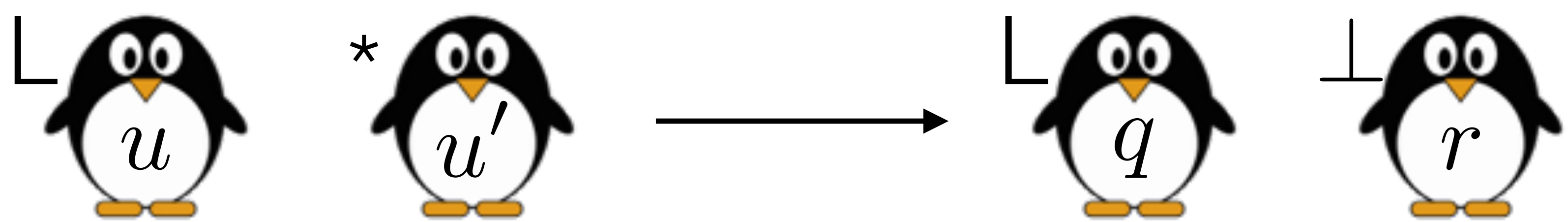
#green - 2 #blue $\leq$ 4

Putting things together



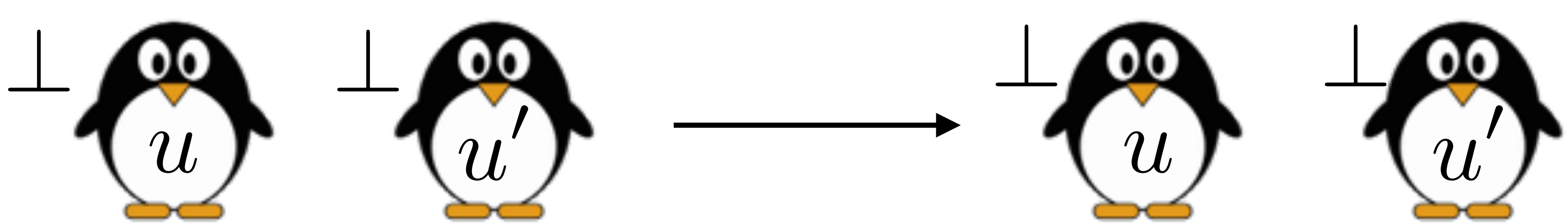
$u_{init} = -2$

$u_{init} = 1$



$$q(u, u') = \max\{-s, \min\{s, u + u'\}\}$$

$$r(u, u') = u + u' - q(u, u')$$



non-leaders copy leader's output

Proof strategy

A. Eventually a single leader

B. Eventually, the leader collects the value

$\max\{-s, \min\{s,$ #  - 2 #  $\}\}$

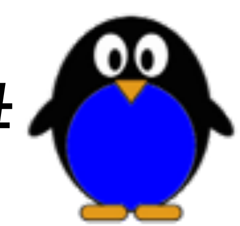
C. Eventually, the agents produce correct outputs

Proof strategy

A. Eventually a single leader

OK

B. Eventually, the leader collects the value

$\max\{-s, \min\{s,$ #  - 2 #  $\}\}$

C. Eventually, the agents produce correct outputs

Proof strategy

A. Eventually a single leader

OK

B. Eventually, the leader collects the value

cf. exercise session

$\max\{-s, \min\{s,$ #  - 2 #  $\}\}$

C. Eventually, the agents produce correct outputs

Proof strategy

C. Eventually, correct outputs

$u_L = \max\{-s, \min\{s,$ #  - 2 #  $\}\}$

Proof strategy

C. Eventually, correct outputs

$u_L = \max\{-s, \min\{s,$ #  - 2 #  $\}\}$

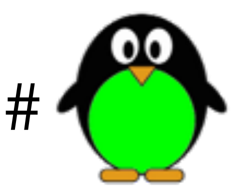
If #  - 2 #  ≤ 4 then $u_L = \begin{cases} \text{\#} \ldots \\ \text{or } -s \end{cases}$ #  - 2 # 

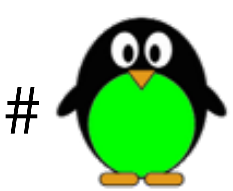
If #  - 2 #  > 4 then $u_L = \begin{cases} \text{\#} \ldots \\ \text{or } s \end{cases}$ #  - 2 # 

Proof strategy

C. Eventually, correct outputs

$u_L = \max\{-s, \min\{s,$ #  - 2 #  $\}\}$

If #  - 2 #  ≤ 4 then $u_L = \begin{cases} \text{\# } \ldots \\ \text{or } -s \end{cases}$ #  - 2 #  or $-s$

If #  - 2 #  > 4 then $u_L = \begin{cases} \text{\# } \ldots \\ \text{or } s \end{cases}$ #  - 2 #  or s

Leader gets correct output

Proof strategy

C. Eventually, correct outputs

$u_L = \max\{-s, \min\{s,$ #  - 2 #  $\}\}$

If #  - 2 #  ≤ 4 then $u_L = \begin{cases} \text{\#} & \\ \text{or } -s & \end{cases}$ #  - 2 #  or $-s$

If #  - 2 #  > 4 then $u_L =$ { #  - 2 #  or s

Leader gets correct output

Others get correct output on meeting the leader

Proof strategy

C. Eventually, correct outputs

$u_L = \max\{-s, \min\{s,$ #  - 2 #  $\}\}$

If #  - 2 #  ≤ 4 then $u_L =$ #  - 2 #  or $-s$

If #  - 2 #  > 4 then $u_L =$ #  - 2 #  or s

QED

Leader gets correct output

Others get correct output on meeting the leader

$$a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k < c$$

$$a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k = c \mod m$$

boolean combinations

Presburger arithmetics

$$a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k < c$$

$$a_1 \cdot \#\sigma_1 + \cdots + a_k \cdot \#\sigma_k = c \mod m$$

boolean combinations

Presburger arithmetics

(beware: integer coefficients)

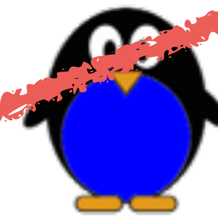
What about multiplication ?

\#  $\cdot$ \#  ≤ 10

What about multiplication ?

~~#  · #  ≤ 10~~ *no!*

What about multiplication ?

~~#  · # ~~ ≤ 10 *no!*

Intuition

1 · #  ≤ 10

2 · #  ≤ 10

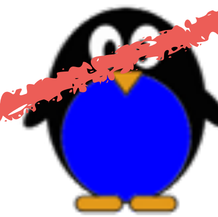
3 · #  ≤ 10

4 · #  ≤ 10

5 · #  ≤ 10

etc.

What about multiplication ?

~~#  · #  ≤ 10~~ no!

Intuition

1 · #  ≤ 10 2 · #  ≤ 10

3 · #  ≤ 10 4 · #  ≤ 10

5 · #  ≤ 10 etc.

previous approach requires too much memory