Playing With Population Protocols

Olivier Bournez[1] Jérémie Chalopin[2,5] Johanne Cohen[3,5]
Xavier Koegler[4] Mikael Rabie[6]

[1]Ecole Polytechnique, France

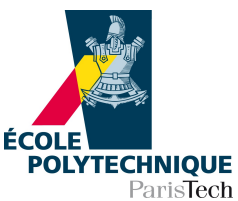
[2]University of Marseille, France

[3]University of Versailles, France

[4]Paris VII University, France

[5]Centre National de la Recherche Scientifique

[6]Ecole Normale Supérieure



Plan

Population Protocols

Variants

Population Protocols and Games
- Symmetric Population Protocols
- Asymmetric Population Protocol

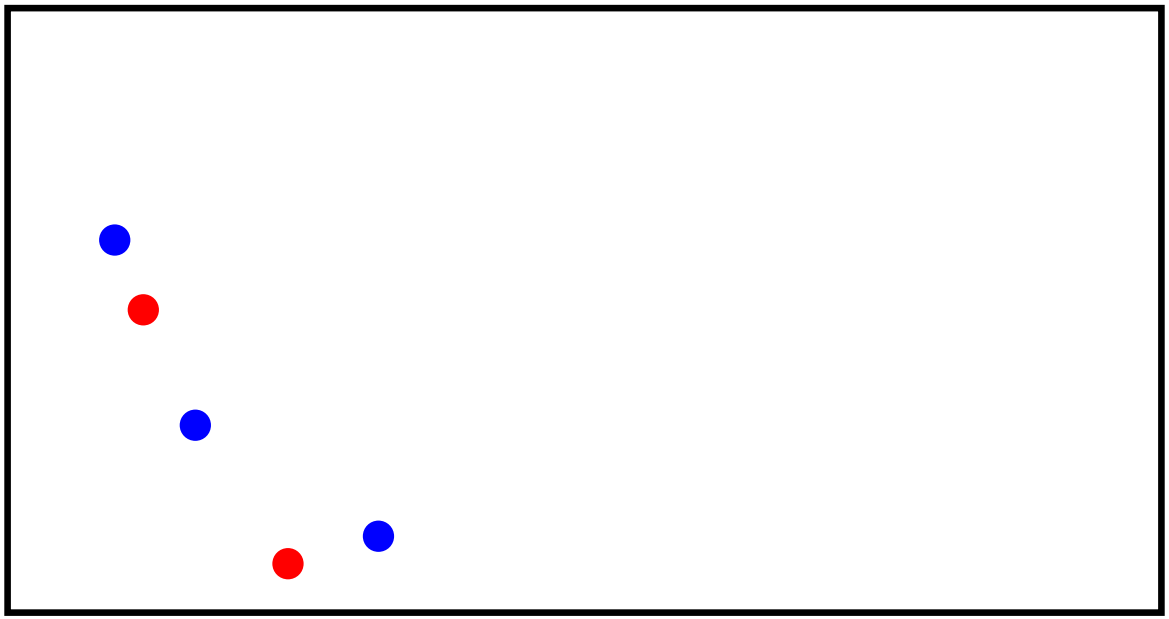
Population protocols

- Introduced by [Angluin,Aspnes,Diamadi,Fischer,Peralta 2004] in the context of distributed systems.

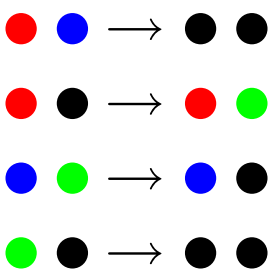
- A model of sensor networks, with absolutely minimal assumptions about

  - sophistication of mobile units:
    - finite state machines.
  - infrastructure: none
    - no topology,
    - not even unique ids.
  - synchrony:
    - totally asynchronous.
  - communications:
    - communications are occasionally possible between pairs of agents.

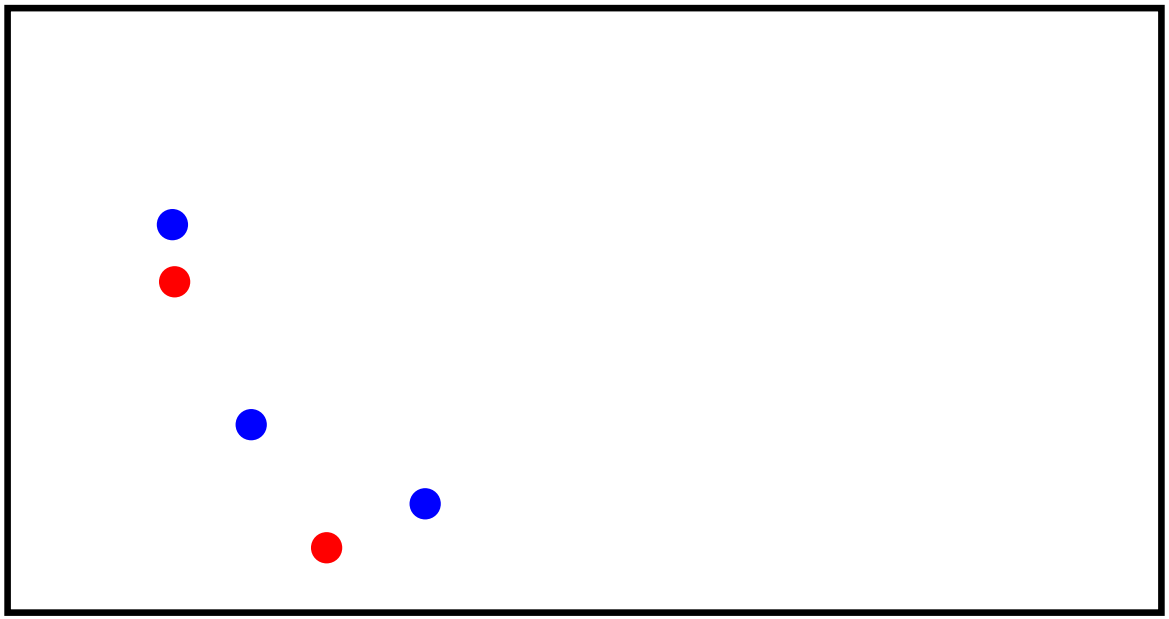
Exemple 1: partant de 3 ●, 2 ●



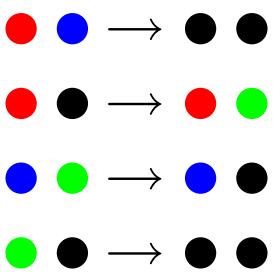
Program:



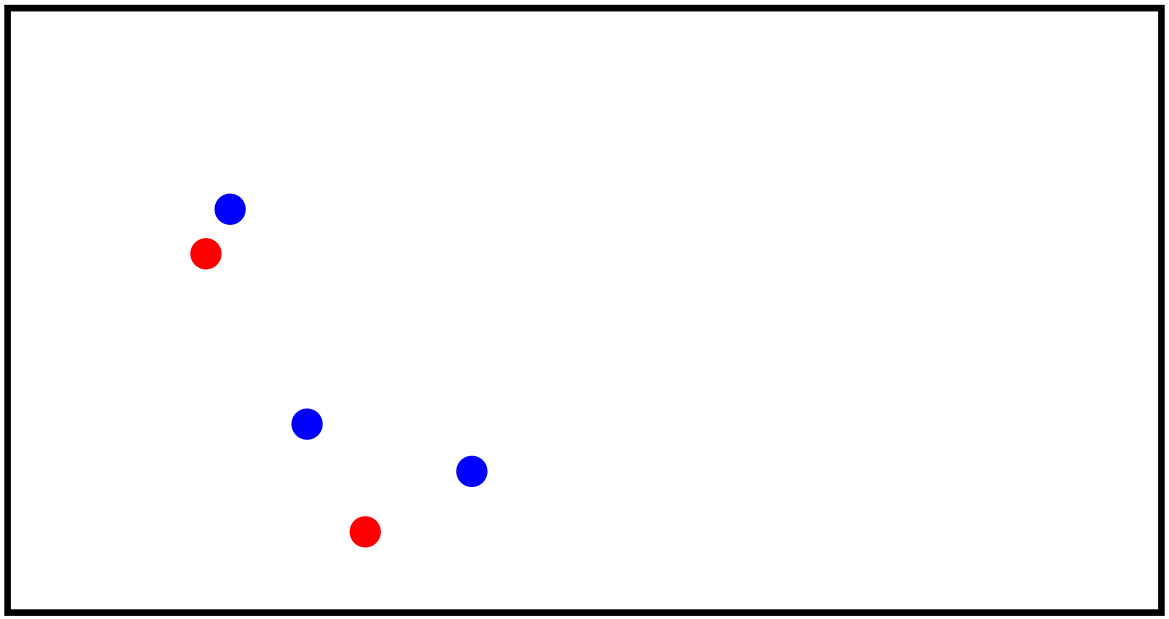
Exemple 1: partant de 3 ●, 2 ●



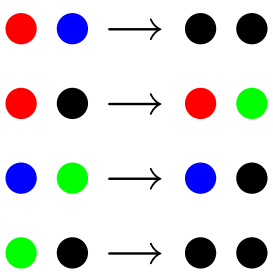
Program:



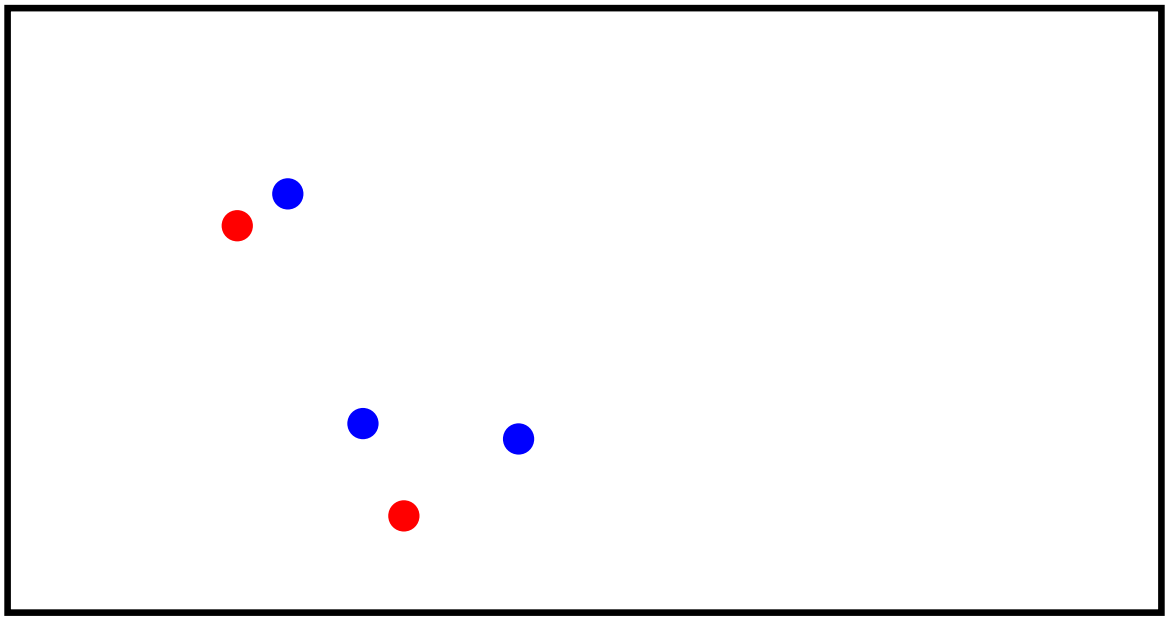
Exemple 1: partant de 3 ●, 2 ●



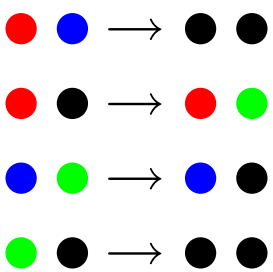
Program:



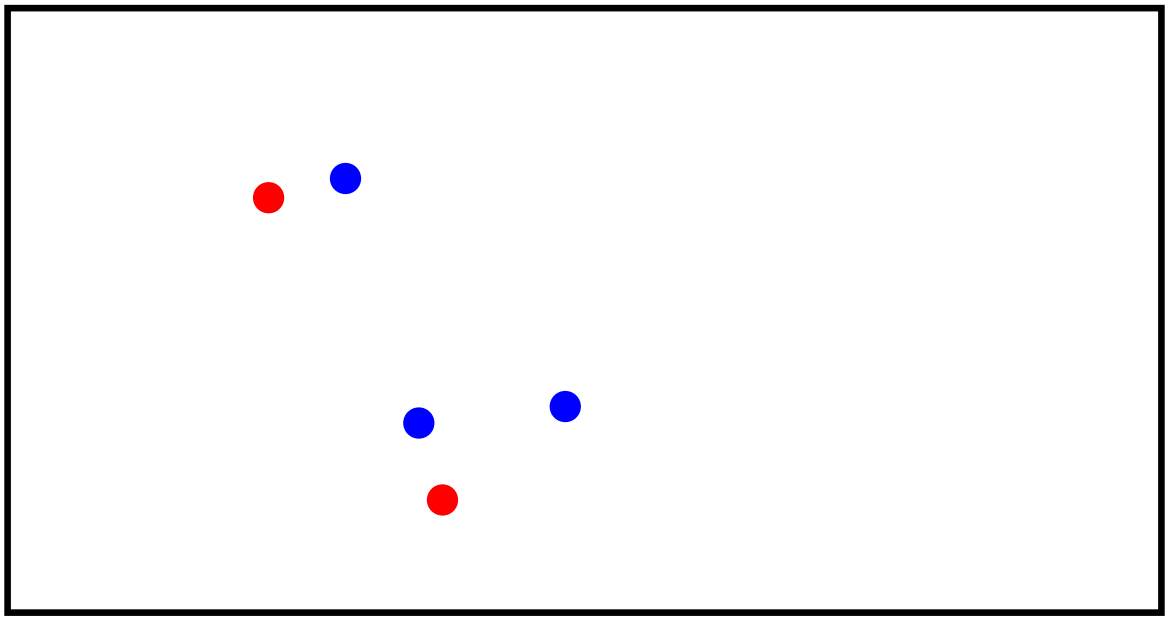
Exemple 1: partant de 3 ●, 2 ●



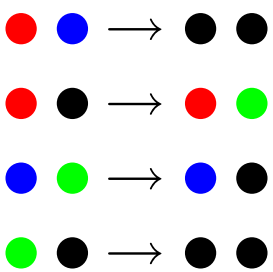
Program:



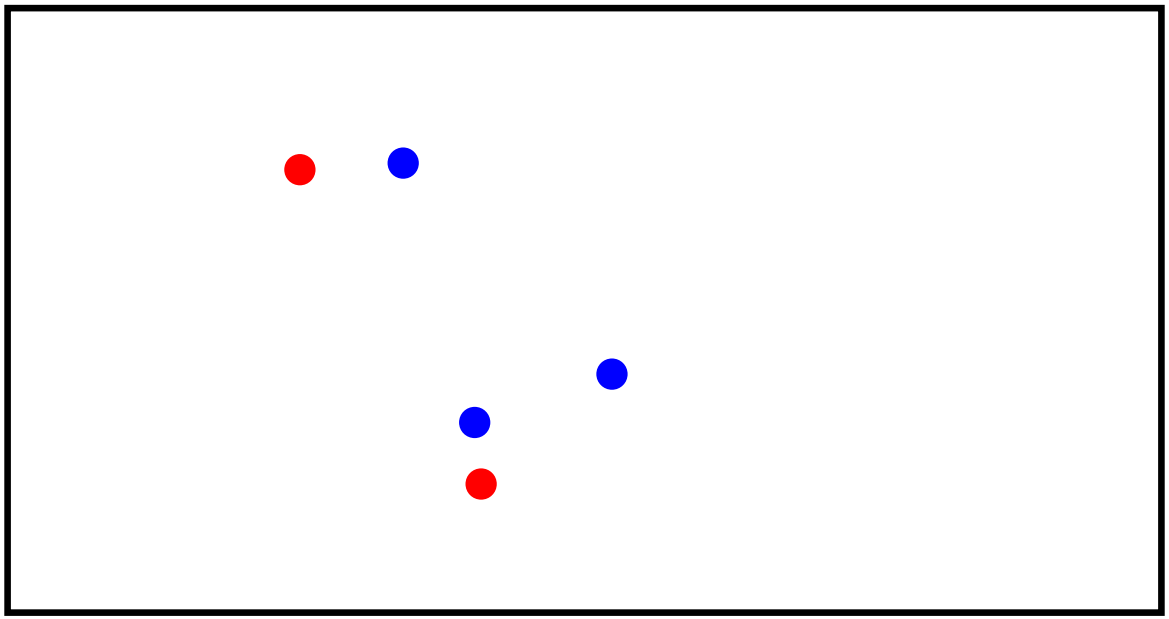
Exemple 1: partant de 3 ●, 2 ●



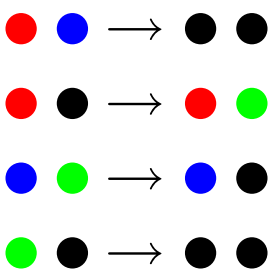
Program:



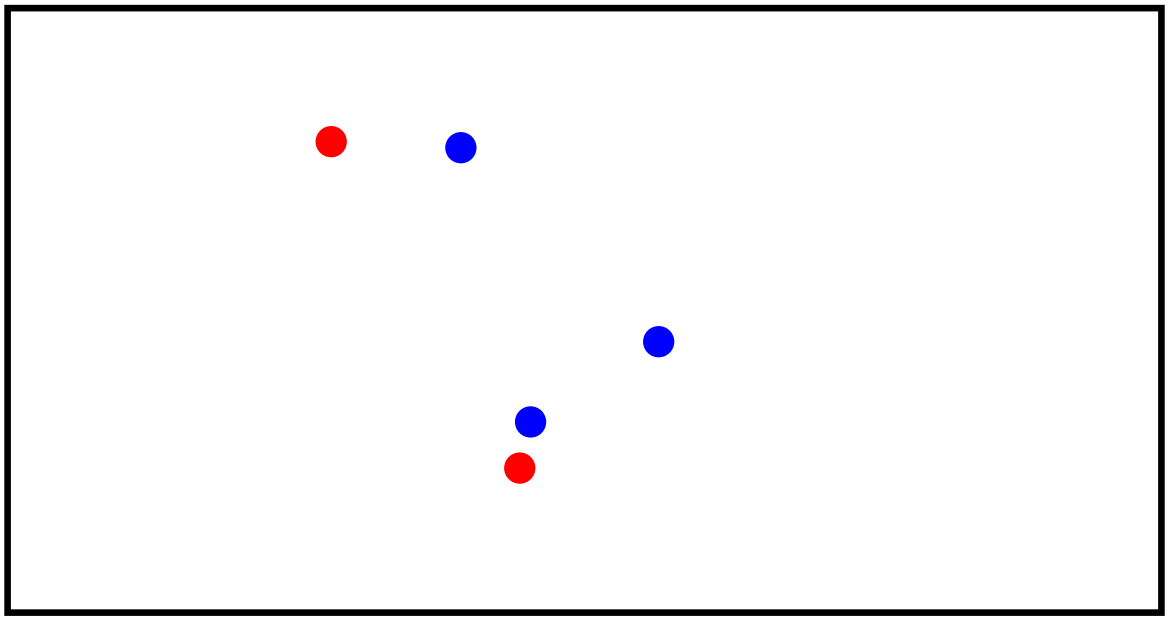
Exemple 1: partant de 3 ●, 2 ●



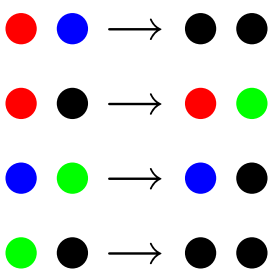
Program:



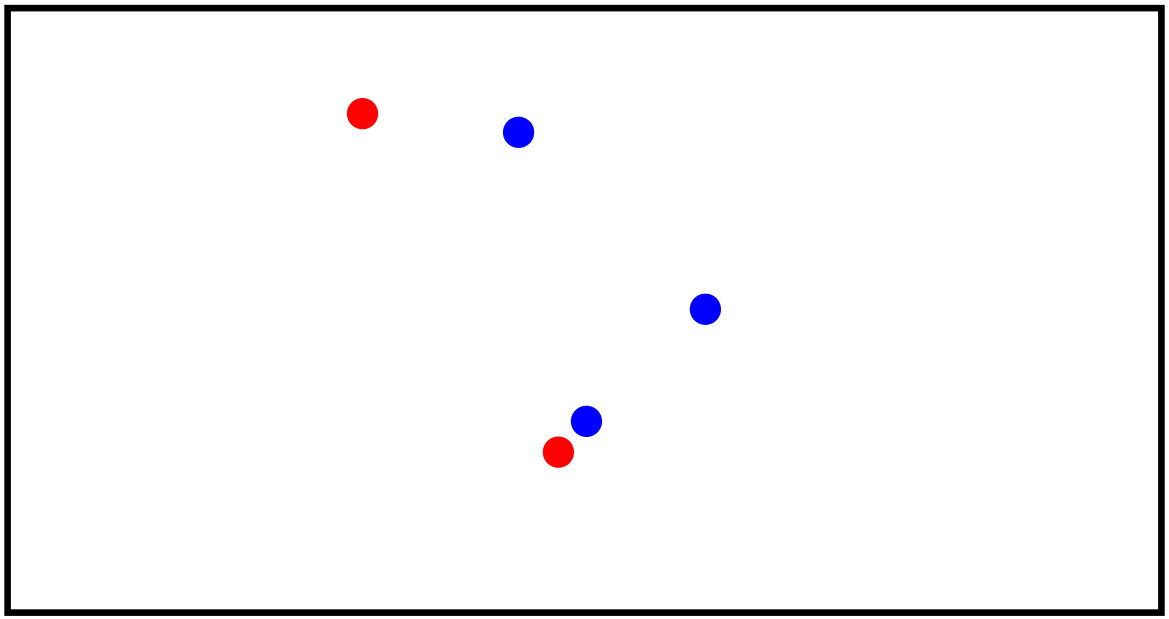
Exemple 1: partant de 3 ●, 2 ●



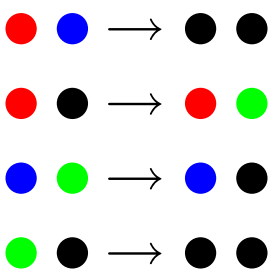
Program:



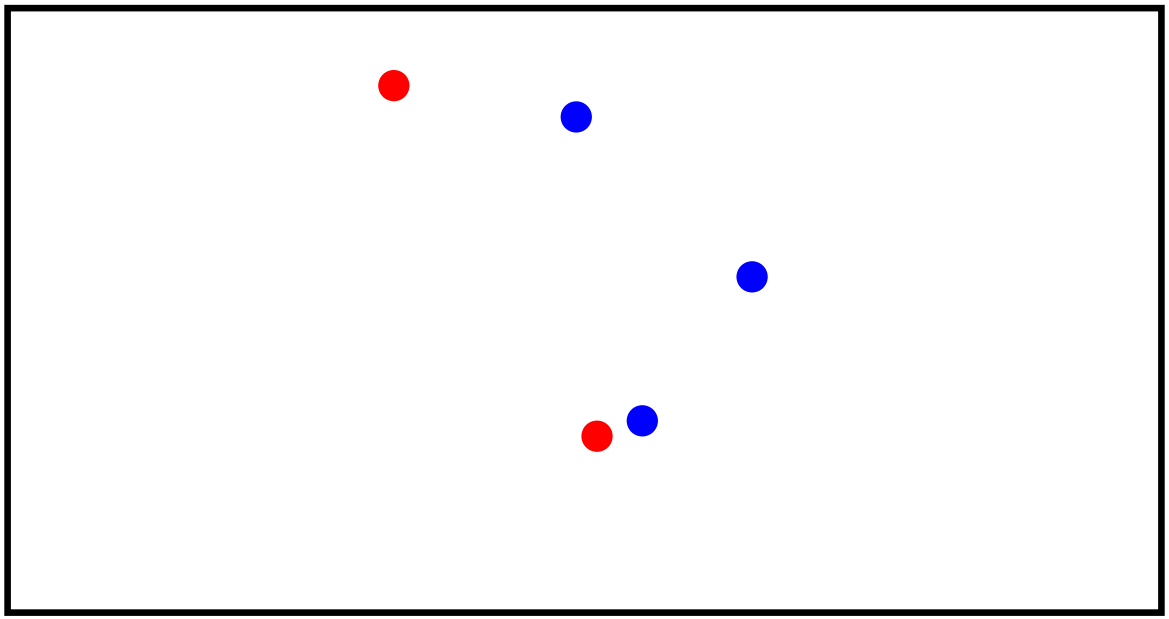
Exemple 1: partant de 3 ●, 2 ●



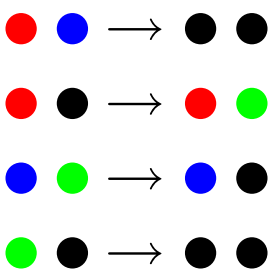
Program:



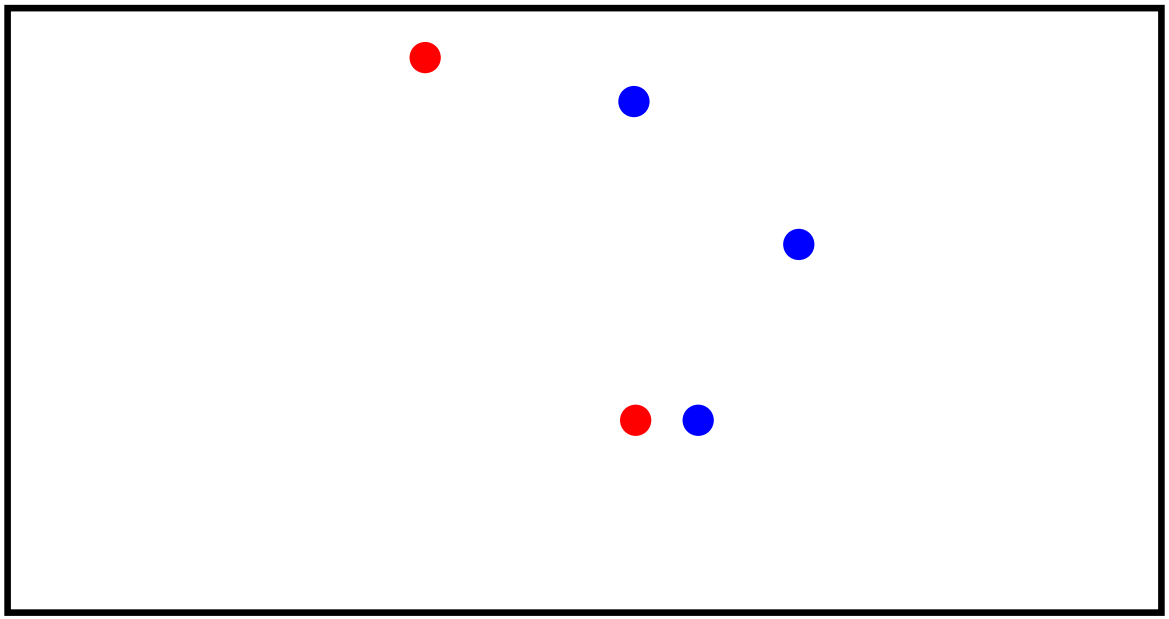
Exemple 1: partant de 3 ●, 2 ●



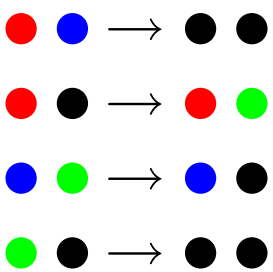
Program:



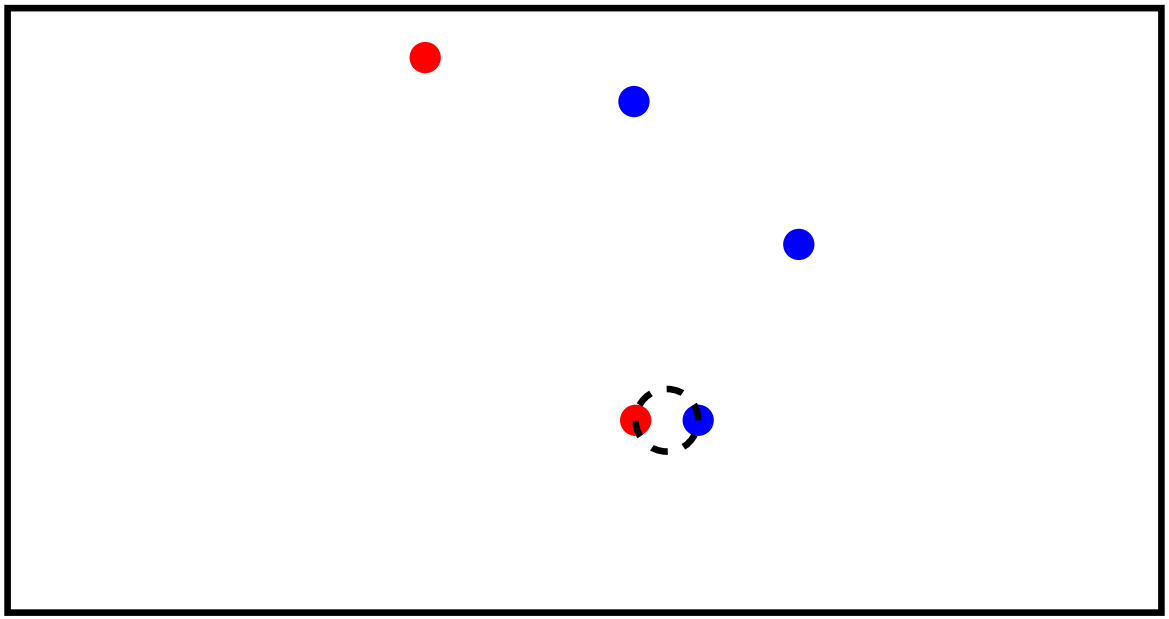
Exemple 1: partant de 3 ●, 2 ●



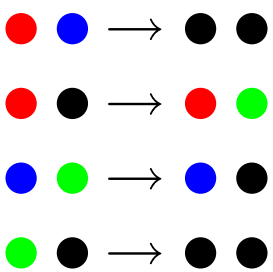
Program:



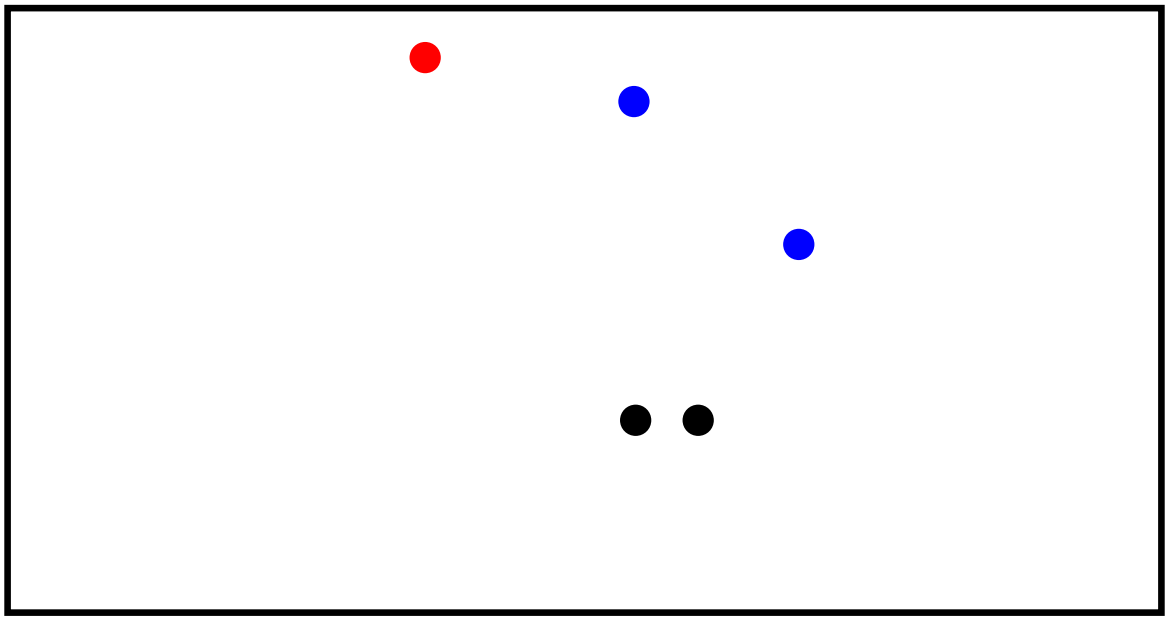
Exemple 1: partant de 3 ●, 2 ●



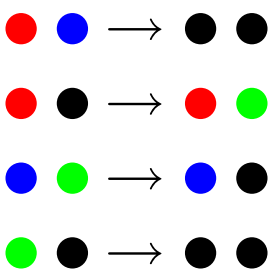
Program:



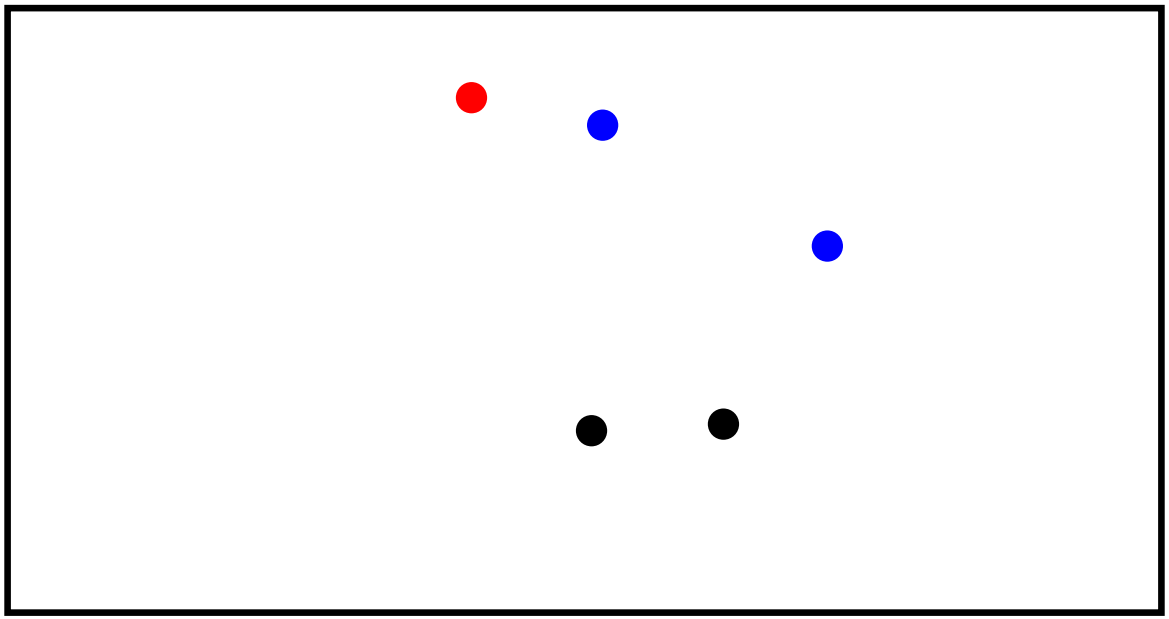
Exemple 1: partant de 3 ●, 2 ●



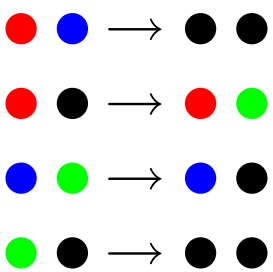
Program:



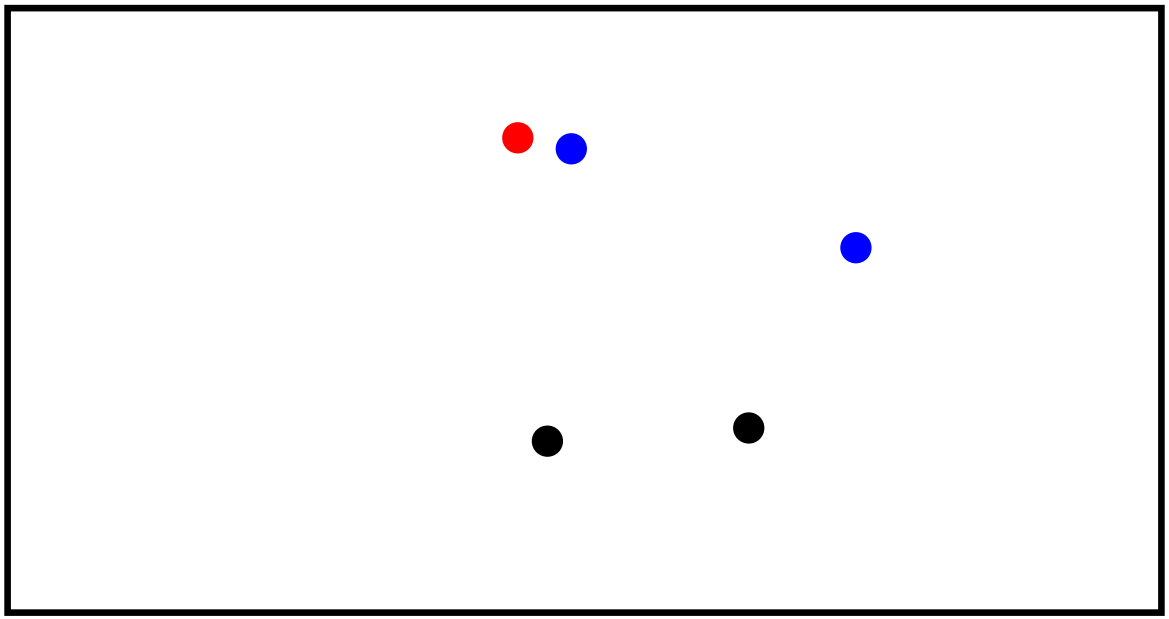
Exemple 1: partant de 3 ●, 2 ●



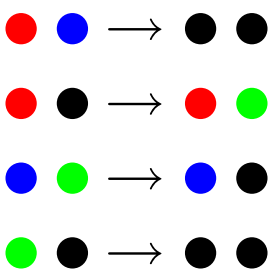
Program:



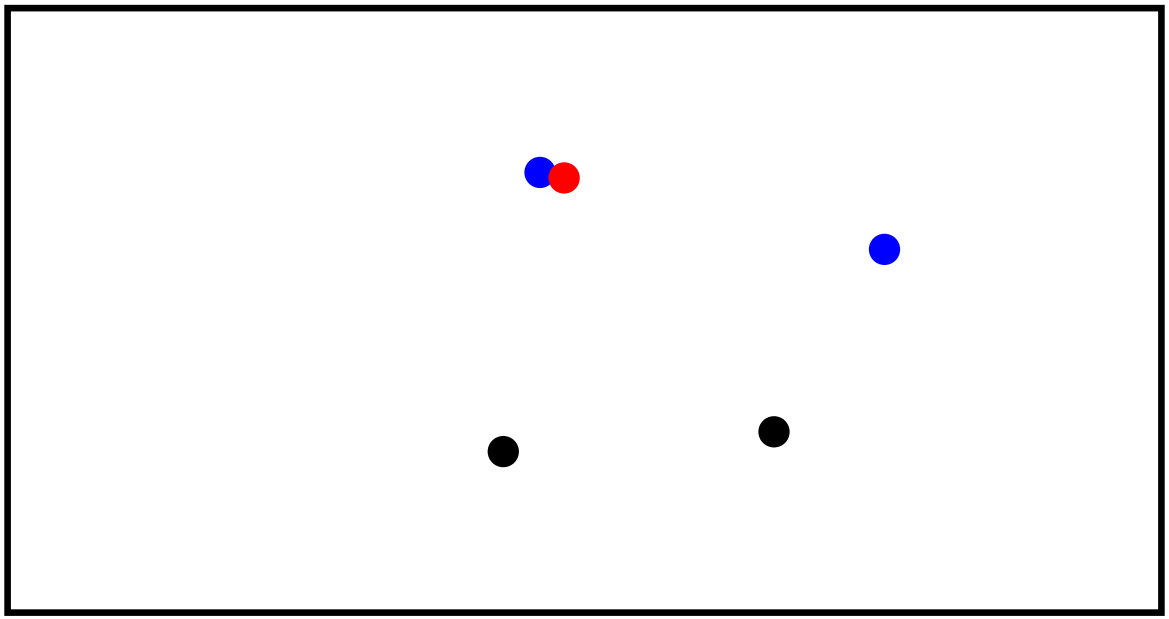
Exemple 1: partant de 3 ●, 2 ●



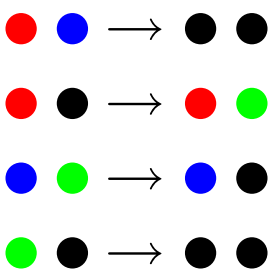
Program:



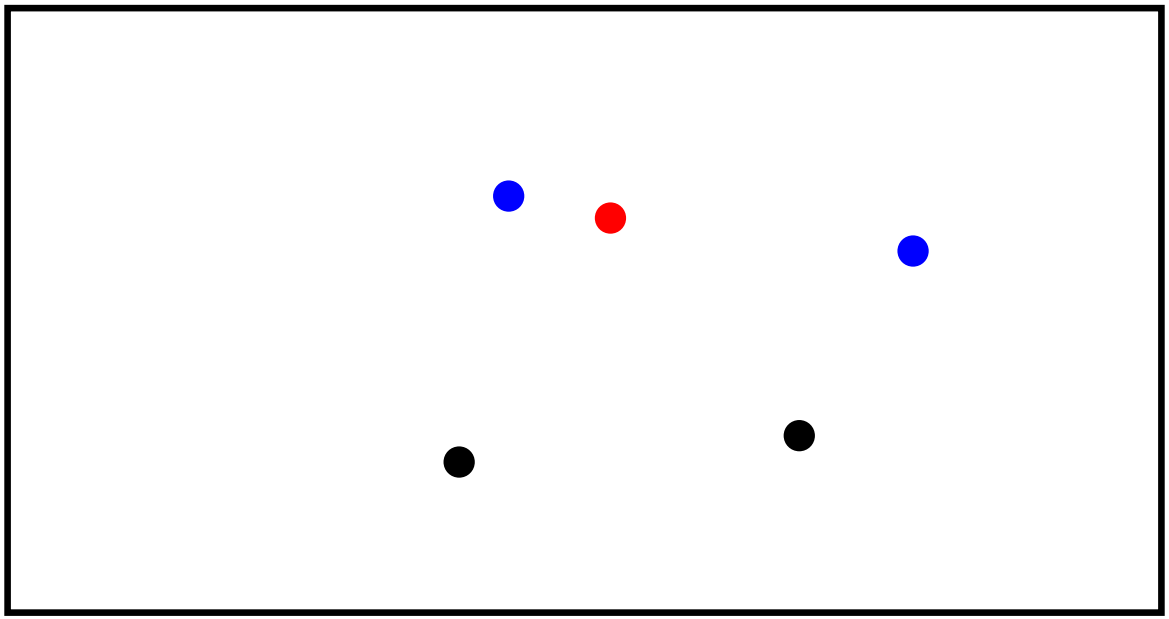
Exemple 1: partant de 3 ●, 2 ●



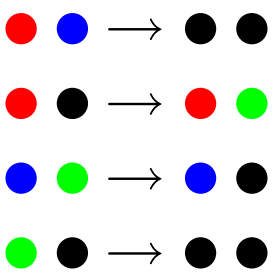
Program:



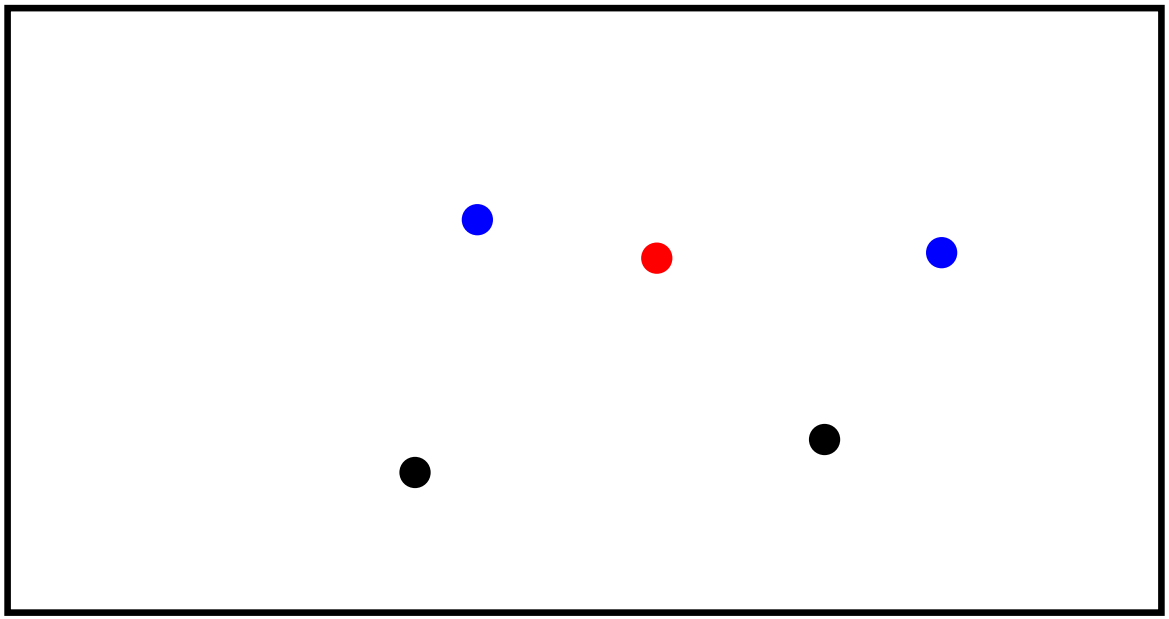
Exemple 1: partant de 3 ●, 2 ●



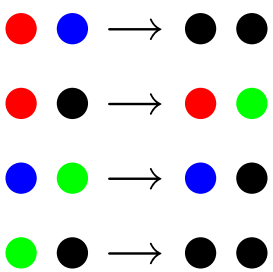
Program:



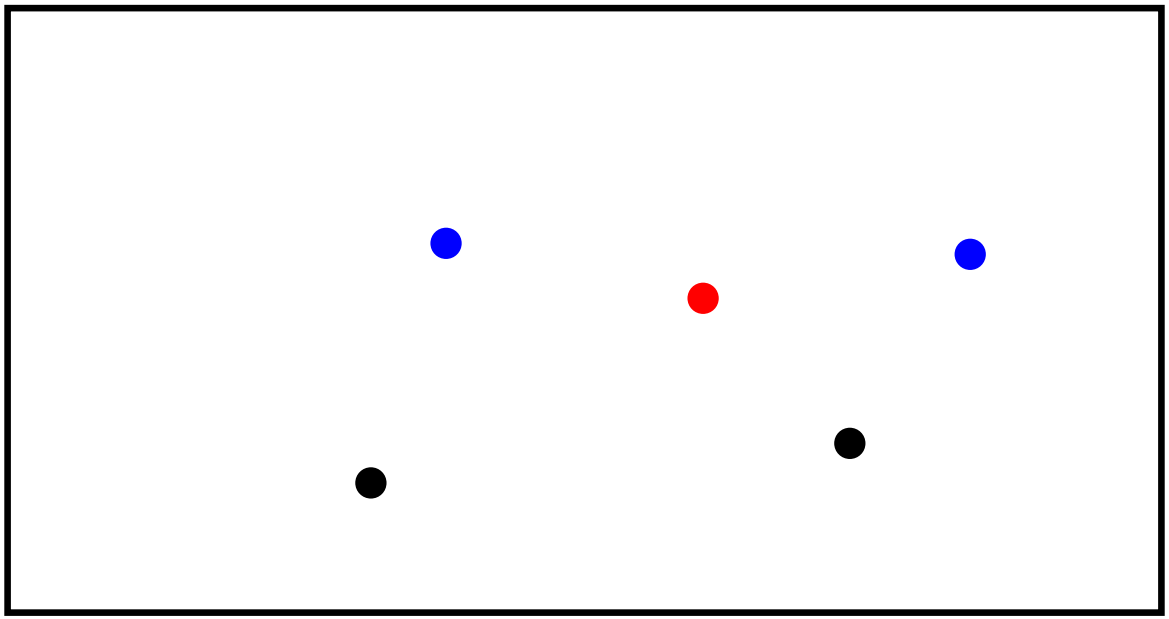
Exemple 1: partant de 3 ●, 2 ●



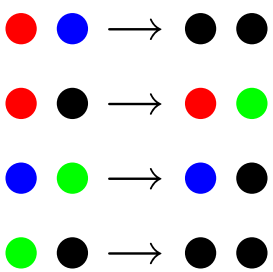
Program:



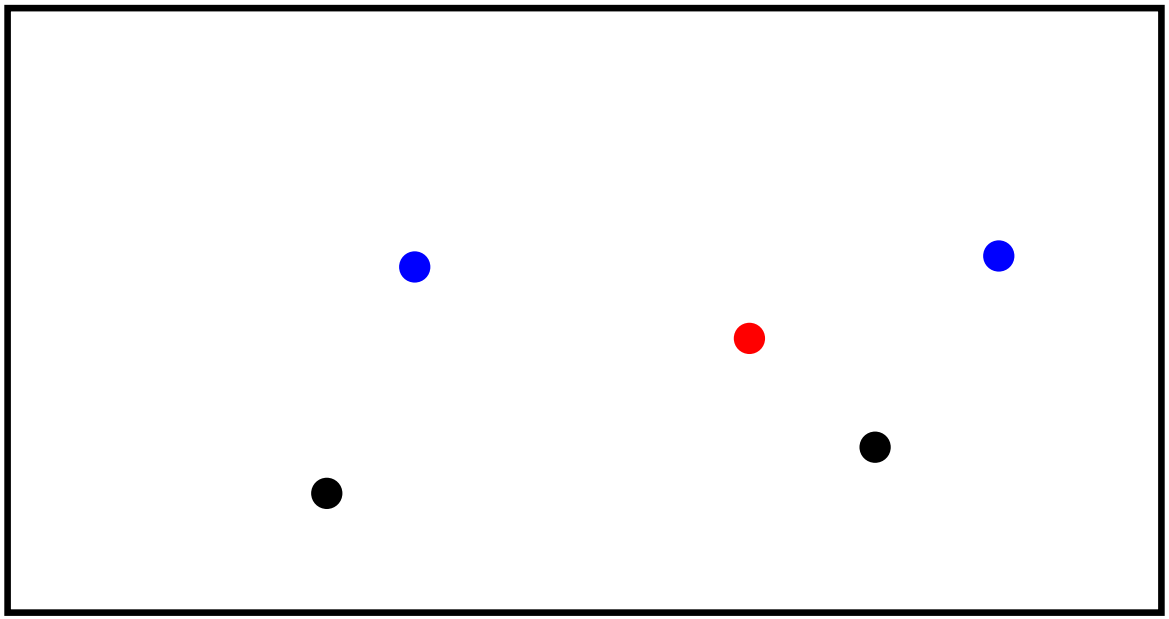
Exemple 1: partant de 3 ●, 2 ●



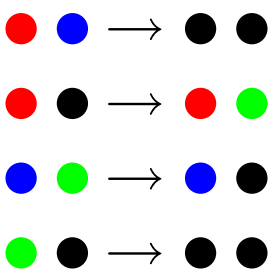
Program:



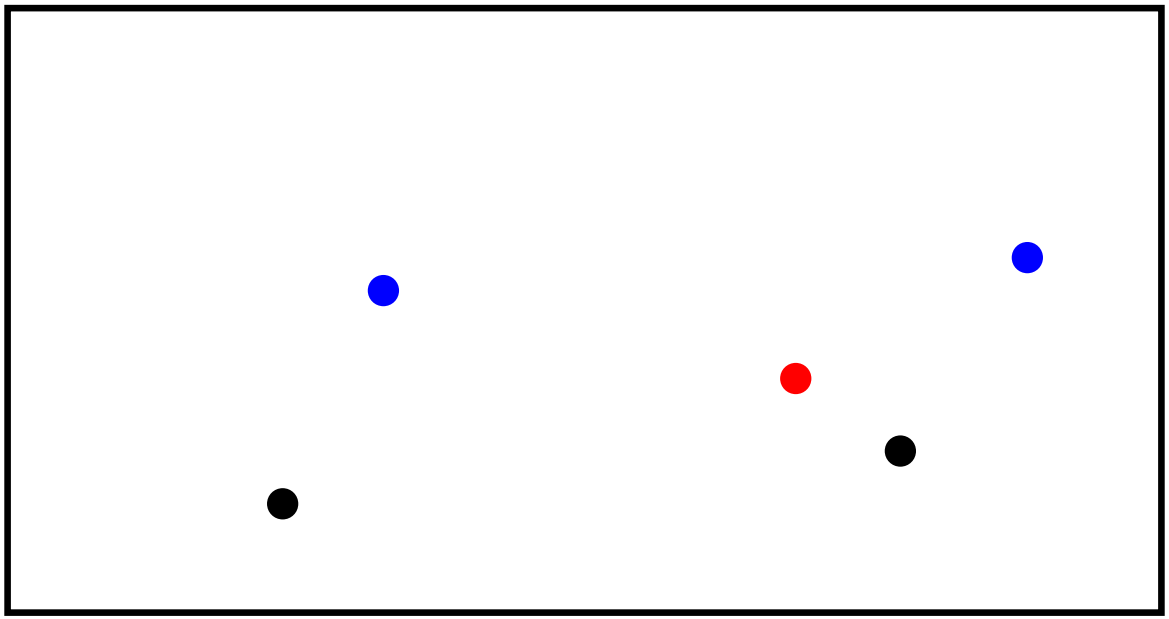
Exemple 1: partant de 3 ●, 2 ●



Program:



Exemple 1: partant de 3 ●, 2 ●



Program:

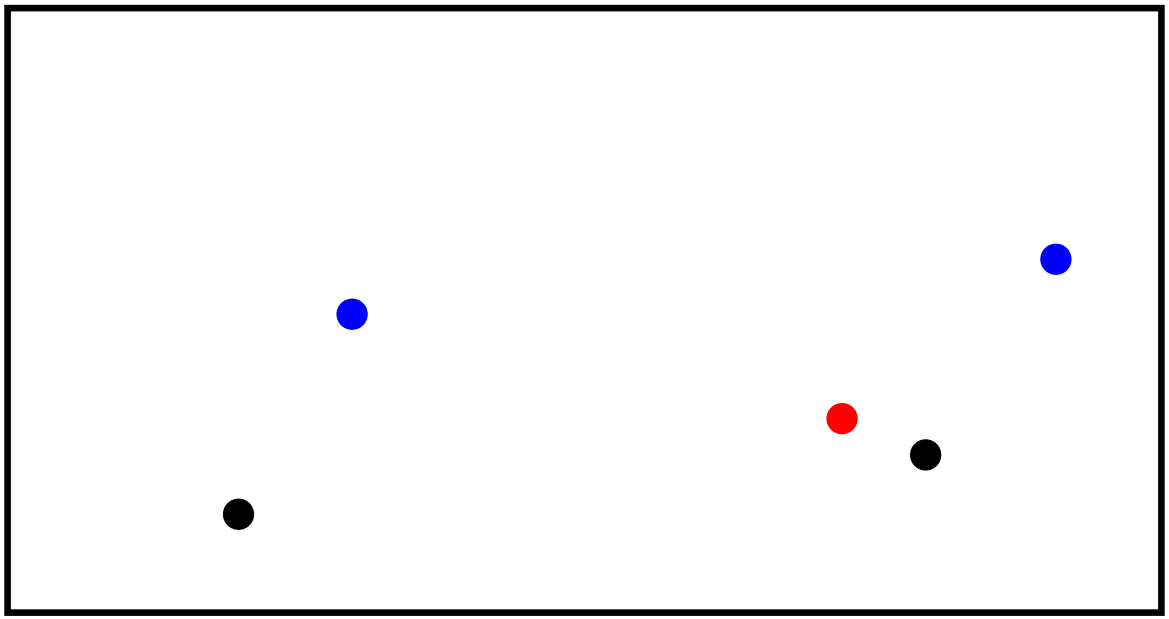
● ● → ● ●

● ● → ● ●

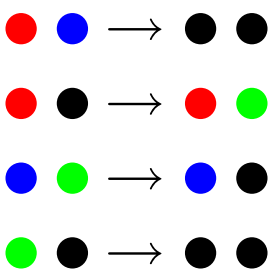
● ● → ● ●

● ● → ● ●

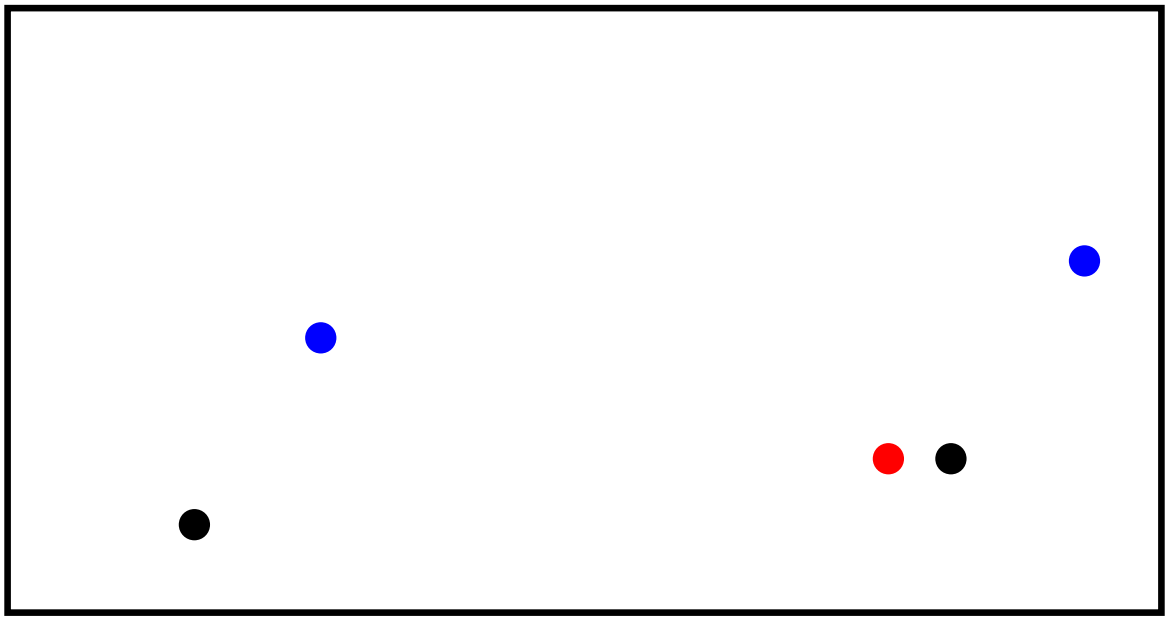
Exemple 1: partant de 3 ●, 2 ●



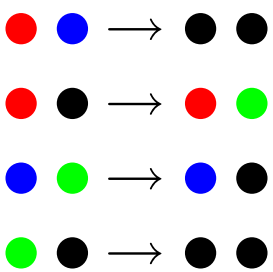
Program:



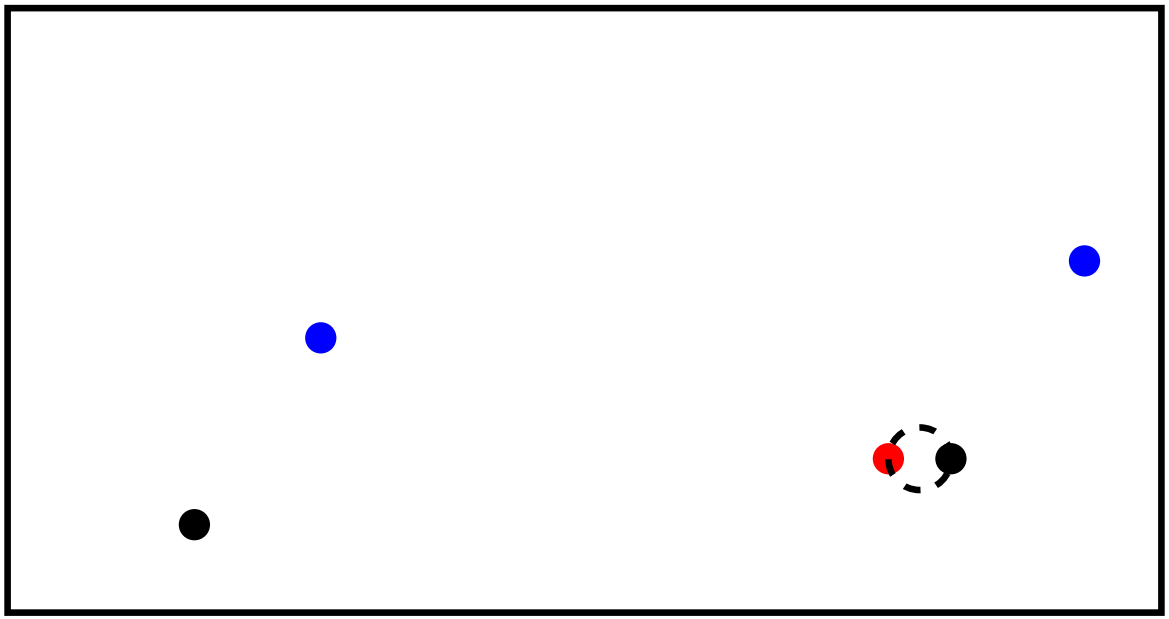
Exemple 1: partant de 3 ●, 2 ●



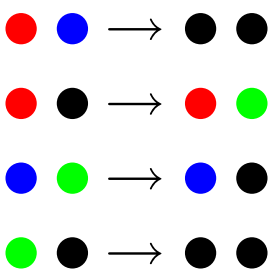
Program:



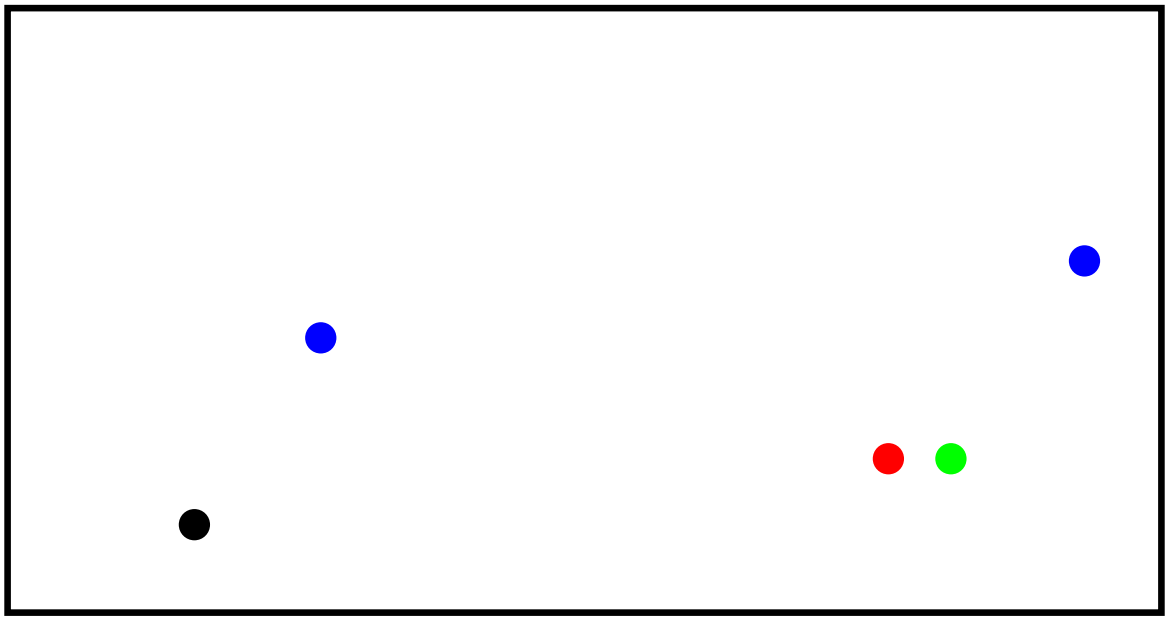
Exemple 1: partant de 3 ●, 2 ●



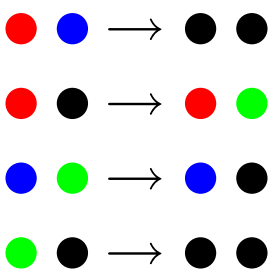
Program:



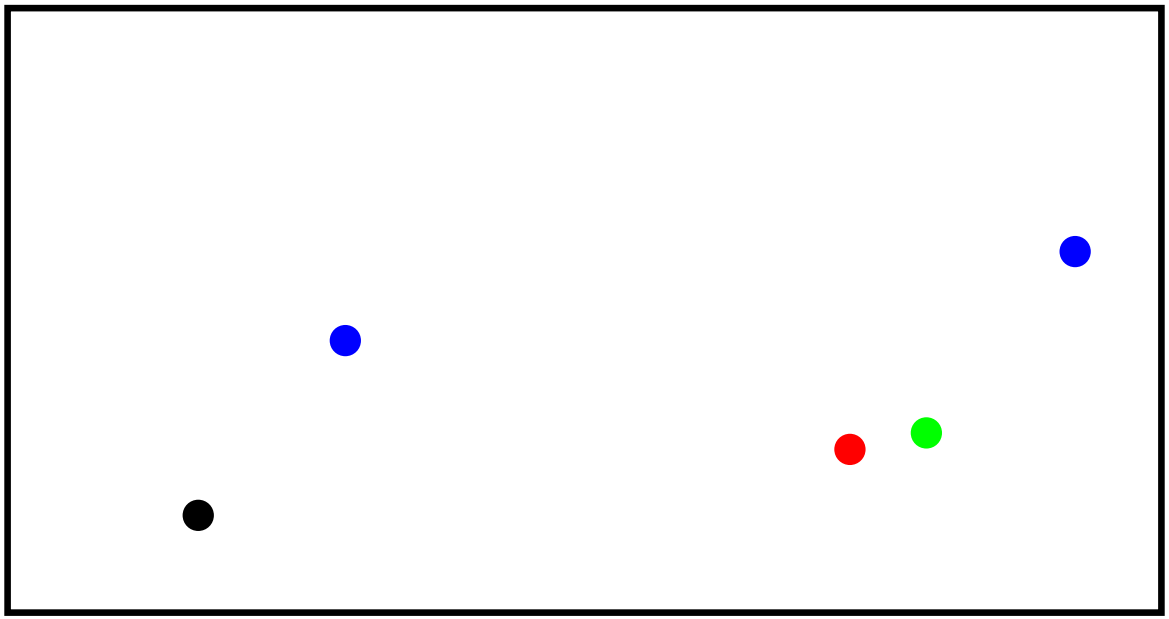
Exemple 1: partant de 3 ●, 2 ●



Program:



Exemple 1: partant de 3 ●, 2 ●



Program:

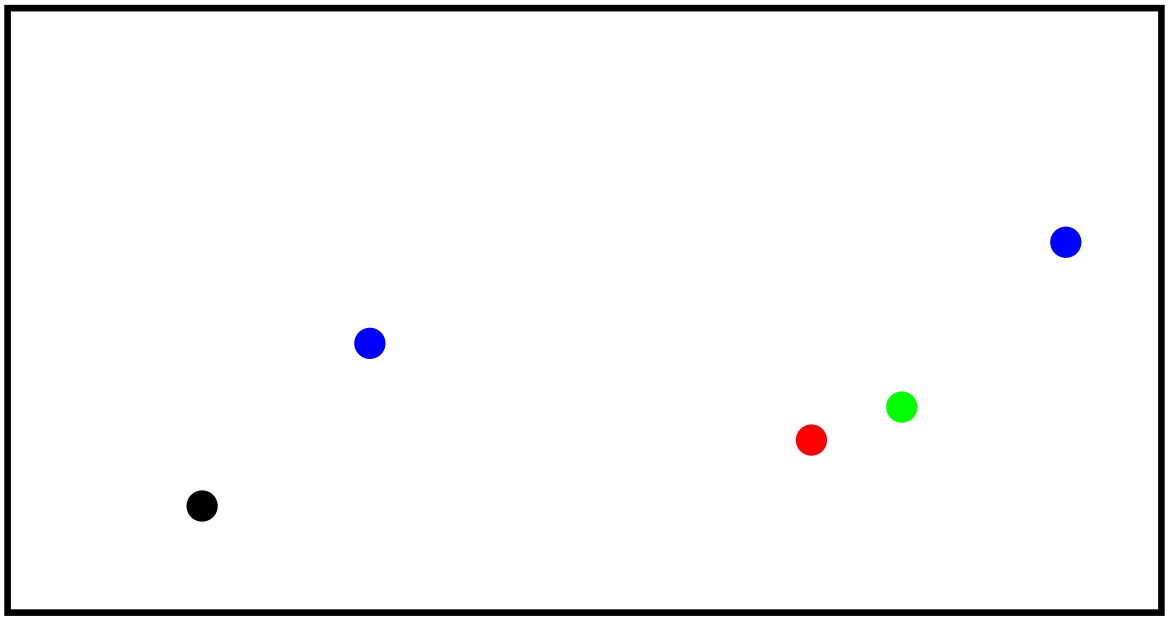
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

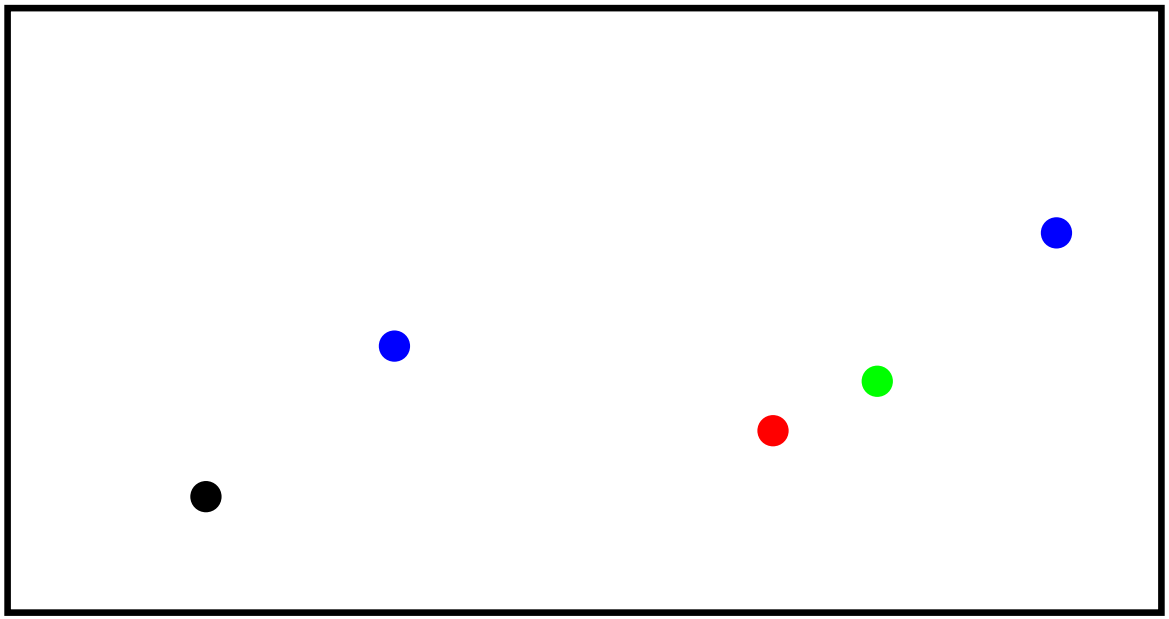
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

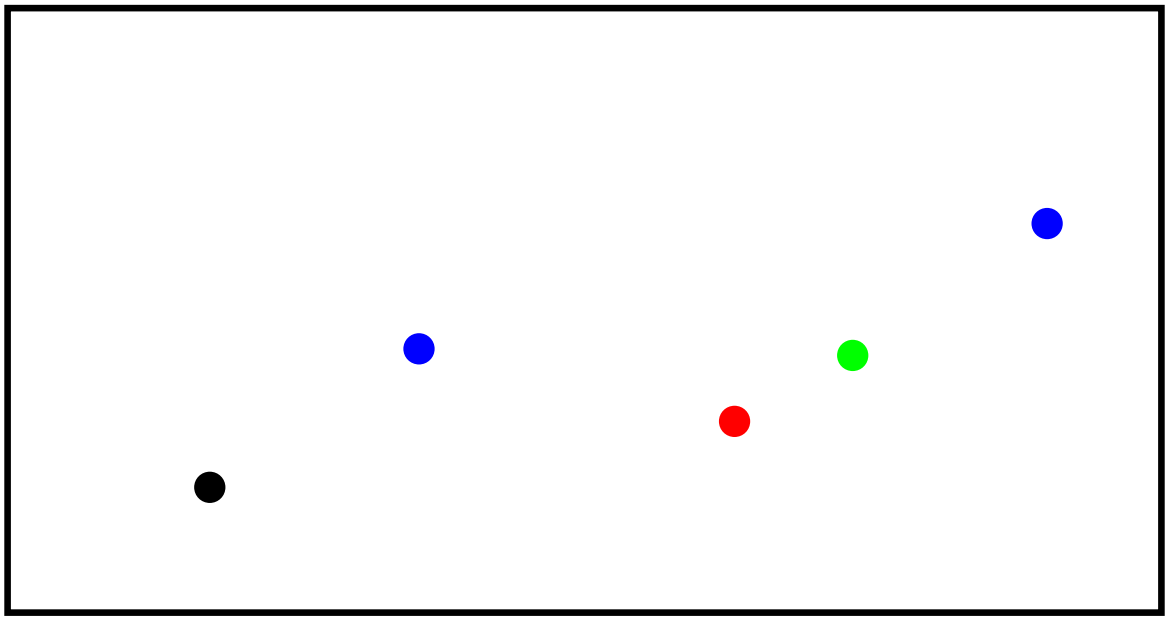
● ● → ● ●

● ● → ● ●

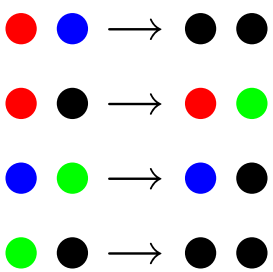
● ● → ● ●

● ● → ● ●

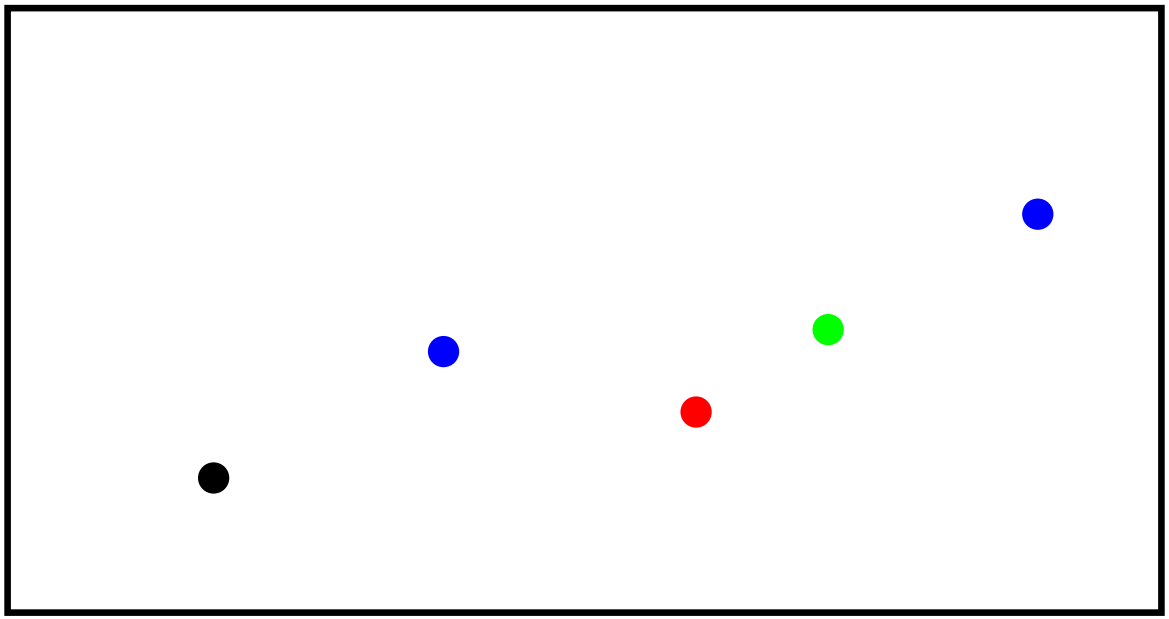
Exemple 1: partant de 3 ●, 2 ●



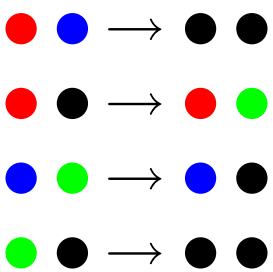
Program:



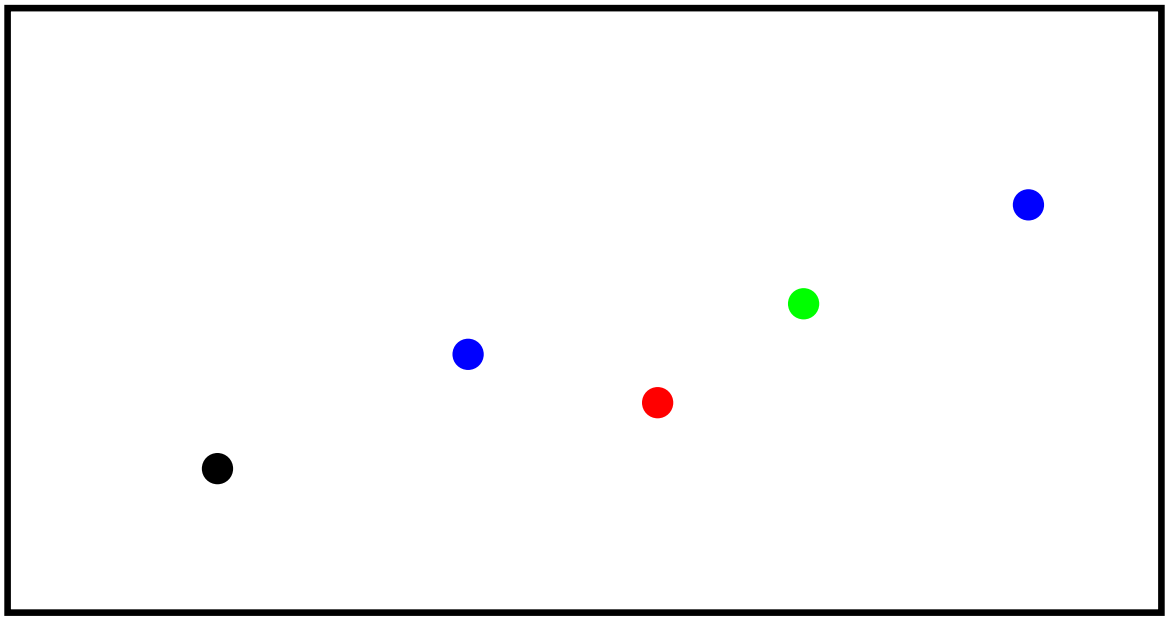
Exemple 1: partant de 3 ●, 2 ●



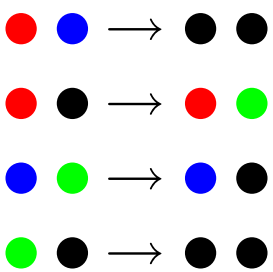
Program:



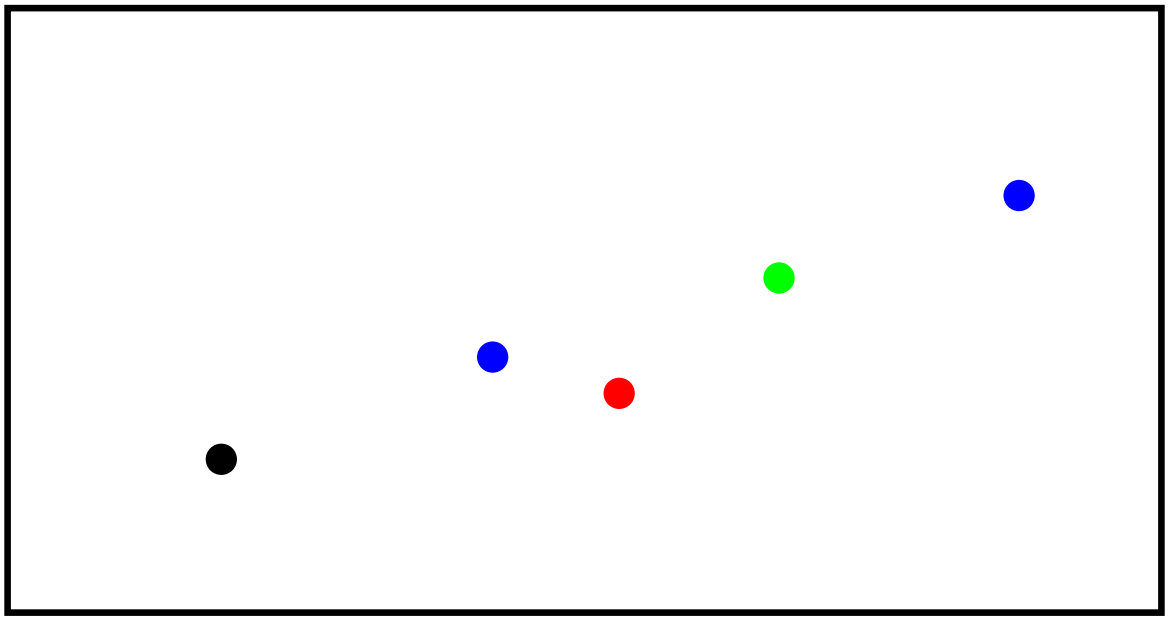
Exemple 1: partant de 3 ●, 2 ●



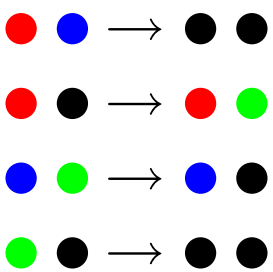
Program:



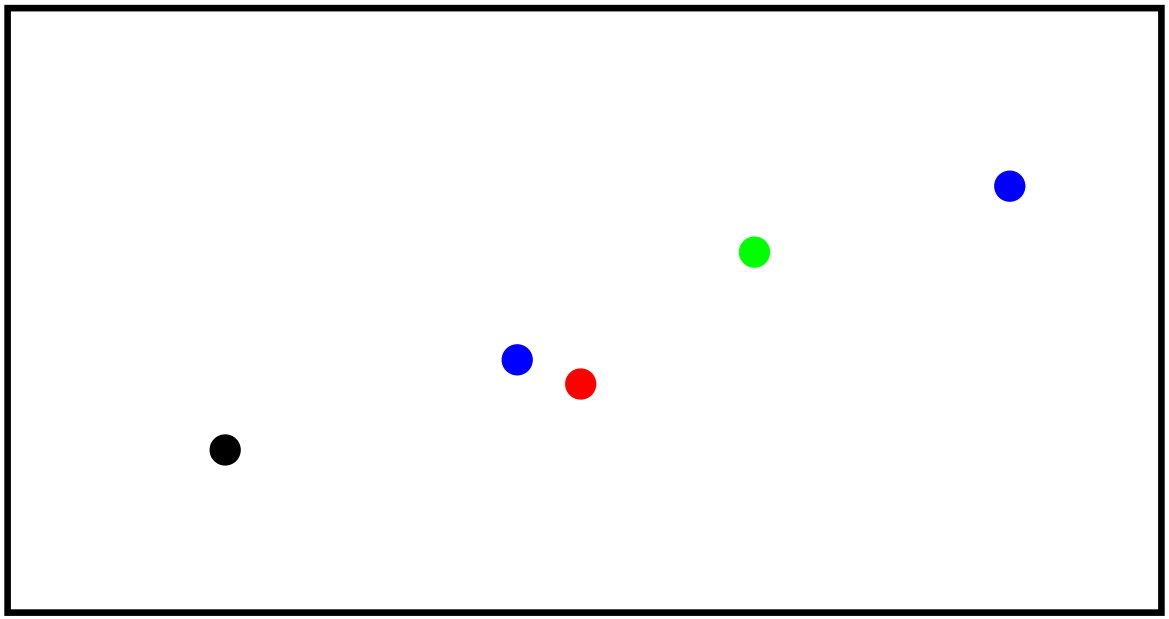
Exemple 1: partant de 3 ●, 2 ●



Program:



Exemple 1: partant de 3 ●, 2 ●



Program:

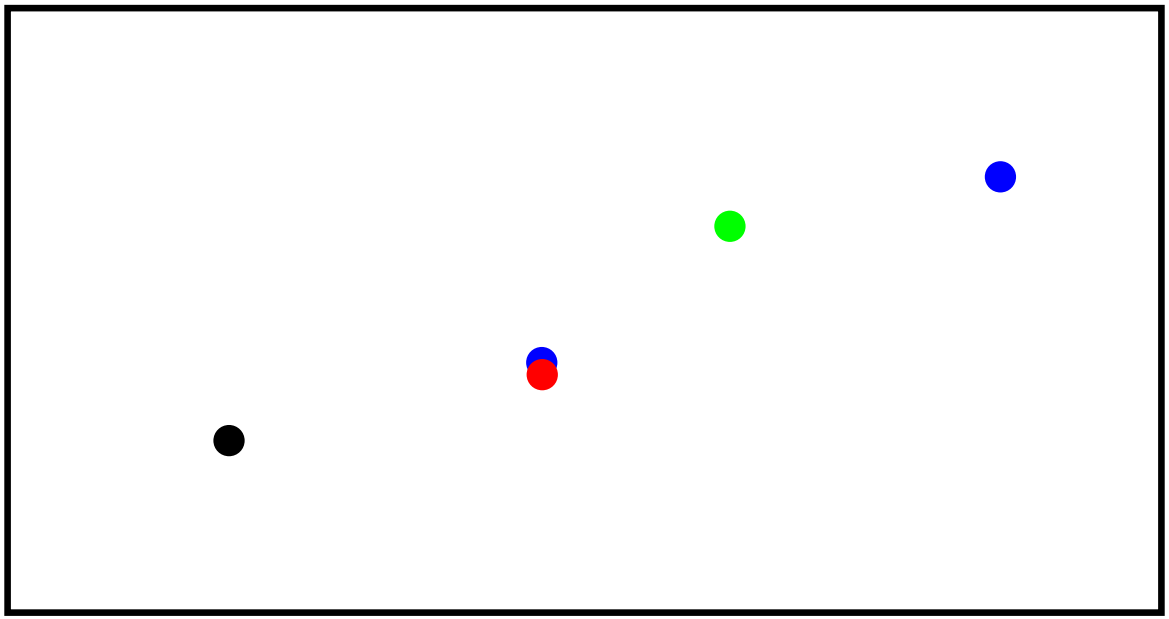
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

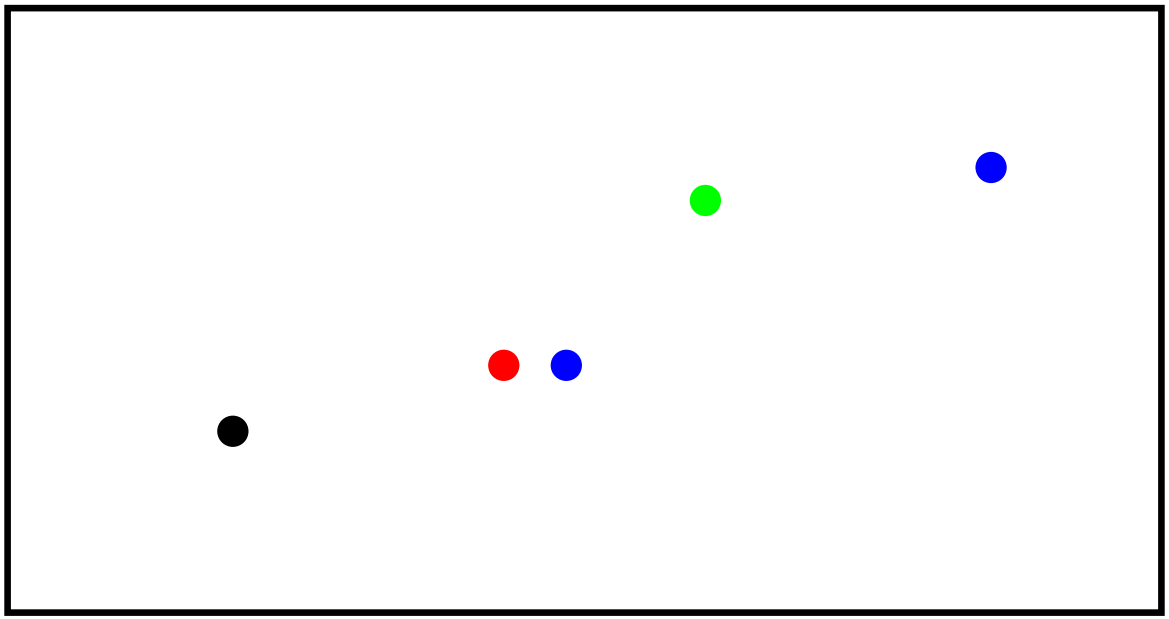
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

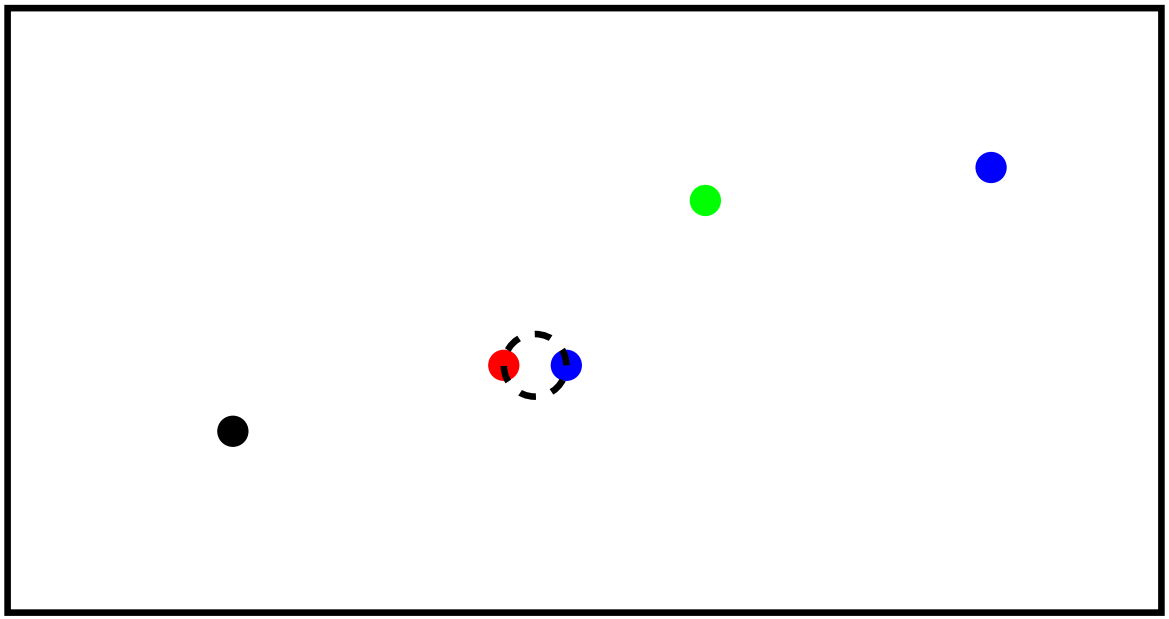
● ● → ● ●

● ● → ● ●

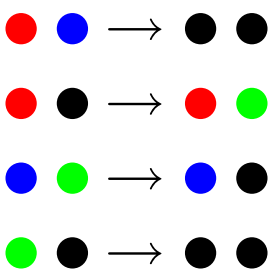
● ● → ● ●

● ● → ● ●

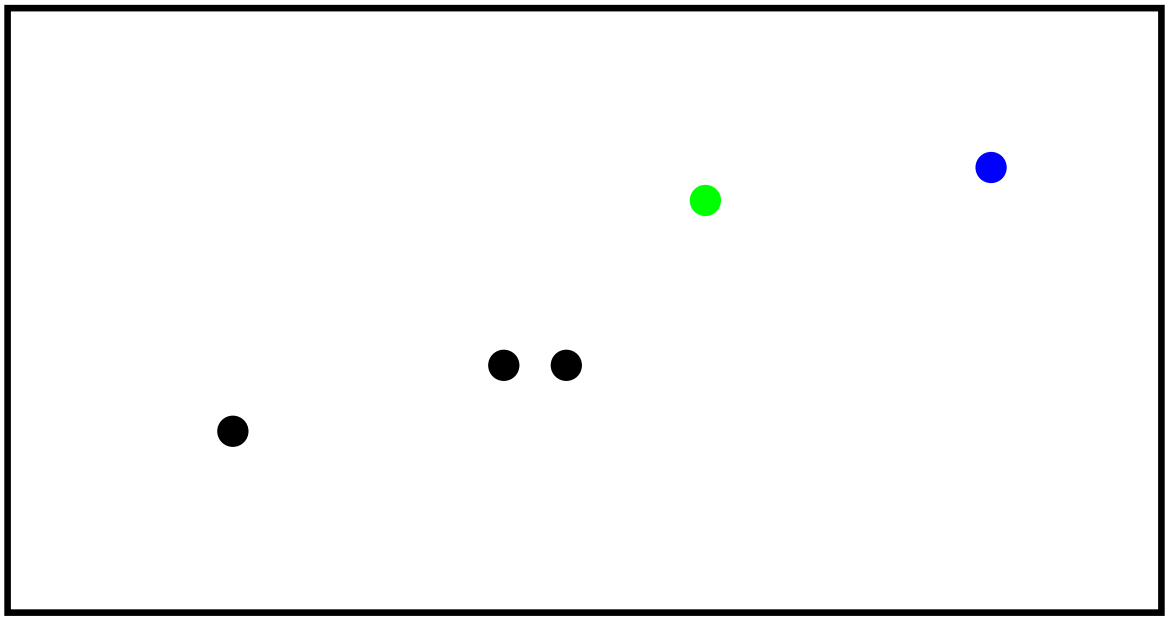
Exemple 1: partant de 3 ●, 2 ●



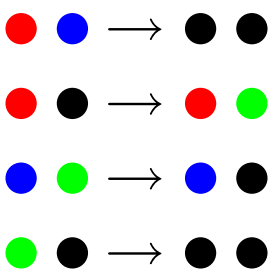
Program:



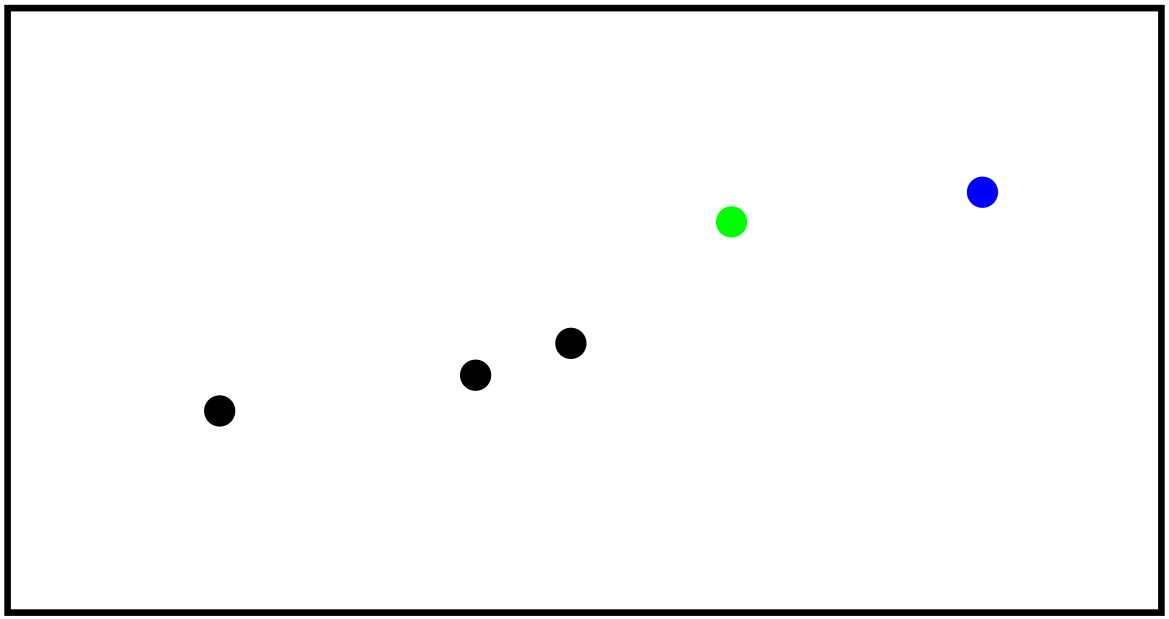
Exemple 1: partant de 3 ●, 2 ●



Program:



Exemple 1: partant de 3 ●, 2 ●



Program:

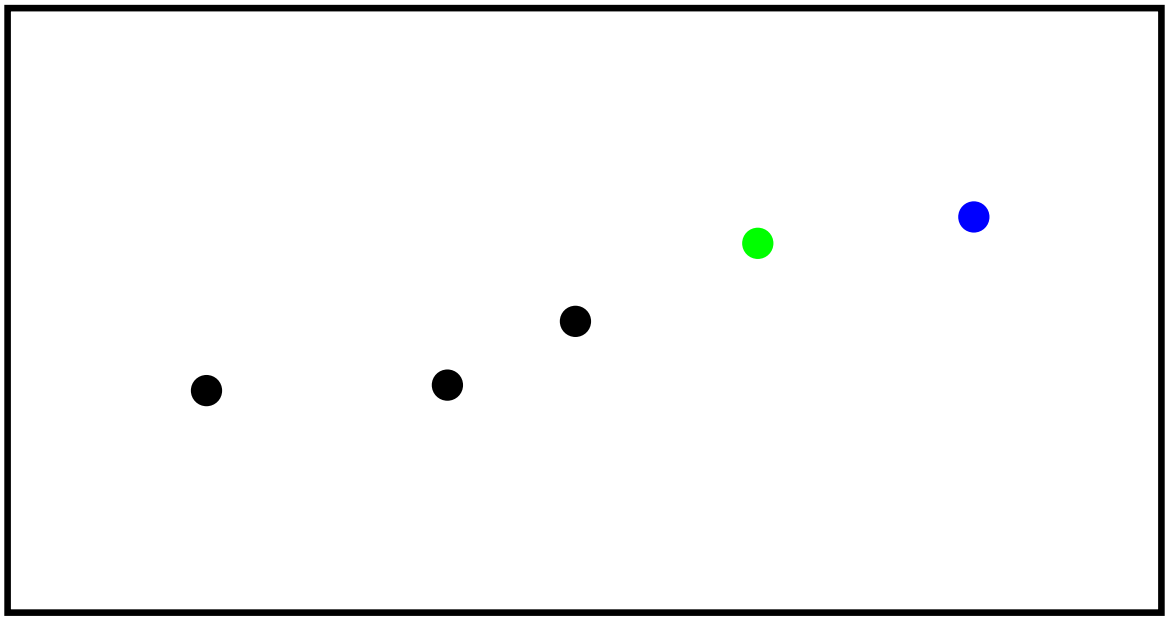
● ● → ● ●

● ● → ● ●

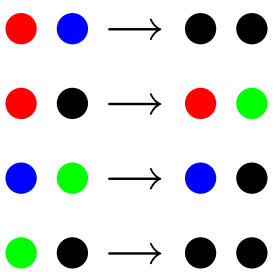
● ● → ● ●

● ● → ● ●

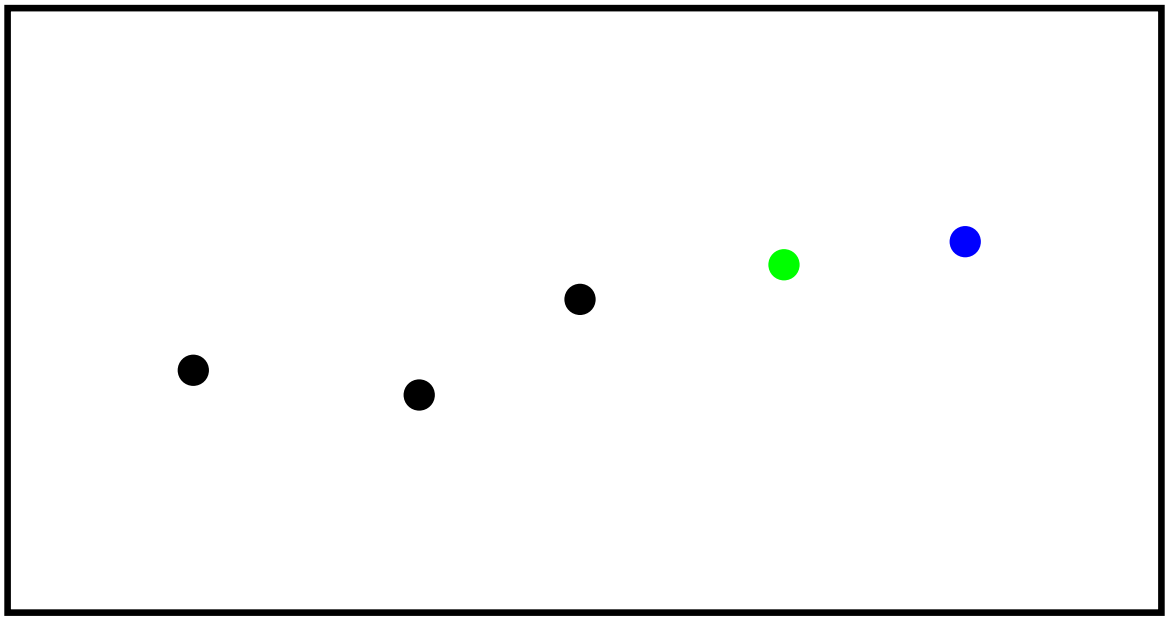
Exemple 1: partant de 3 ●, 2 ●



Program:



Exemple 1: partant de 3 ●, 2 ●



Program:

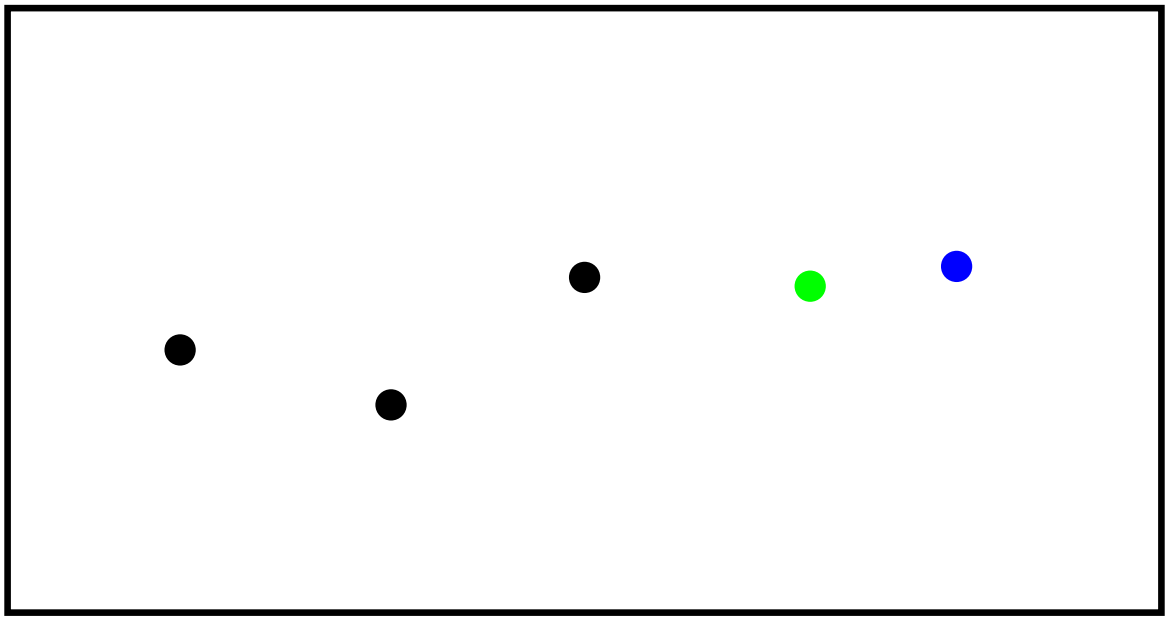
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

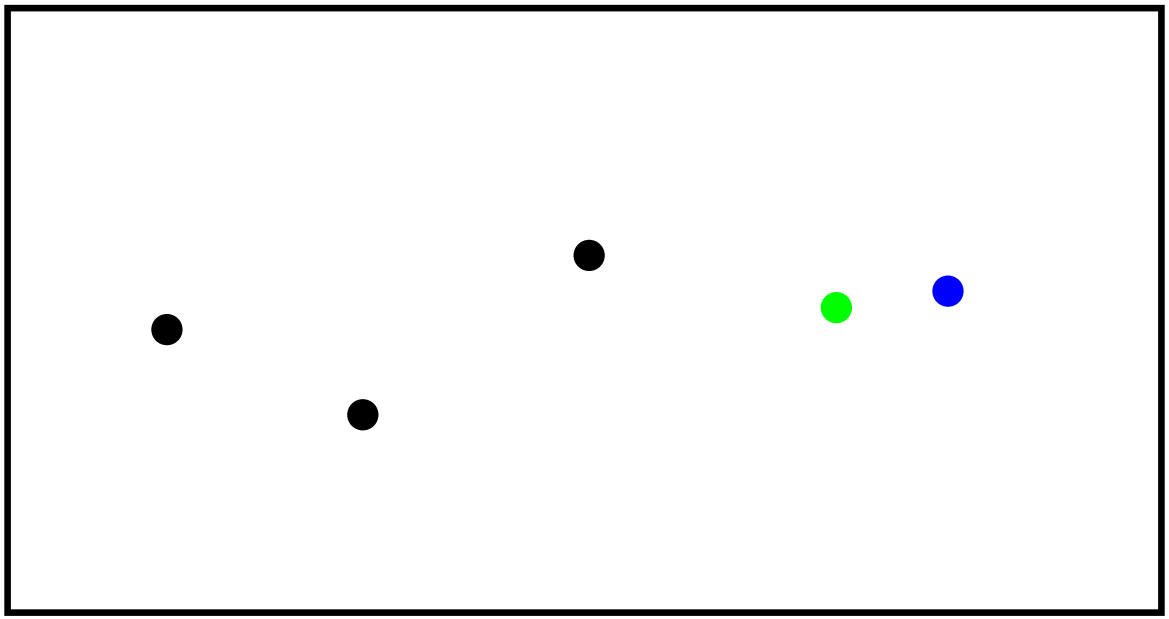
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

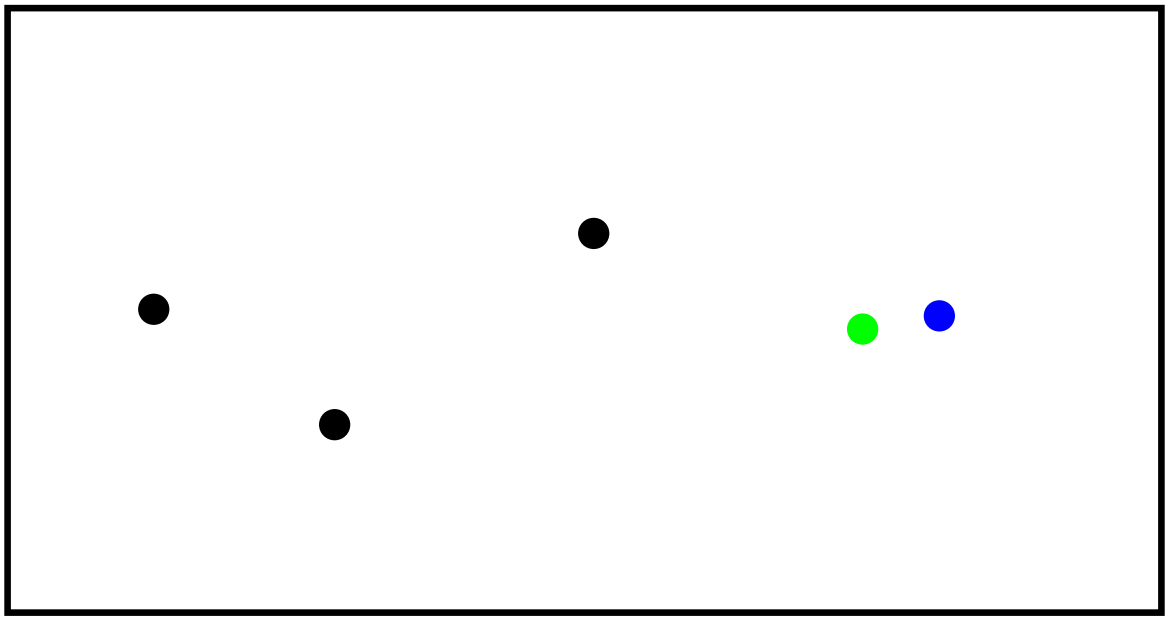
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

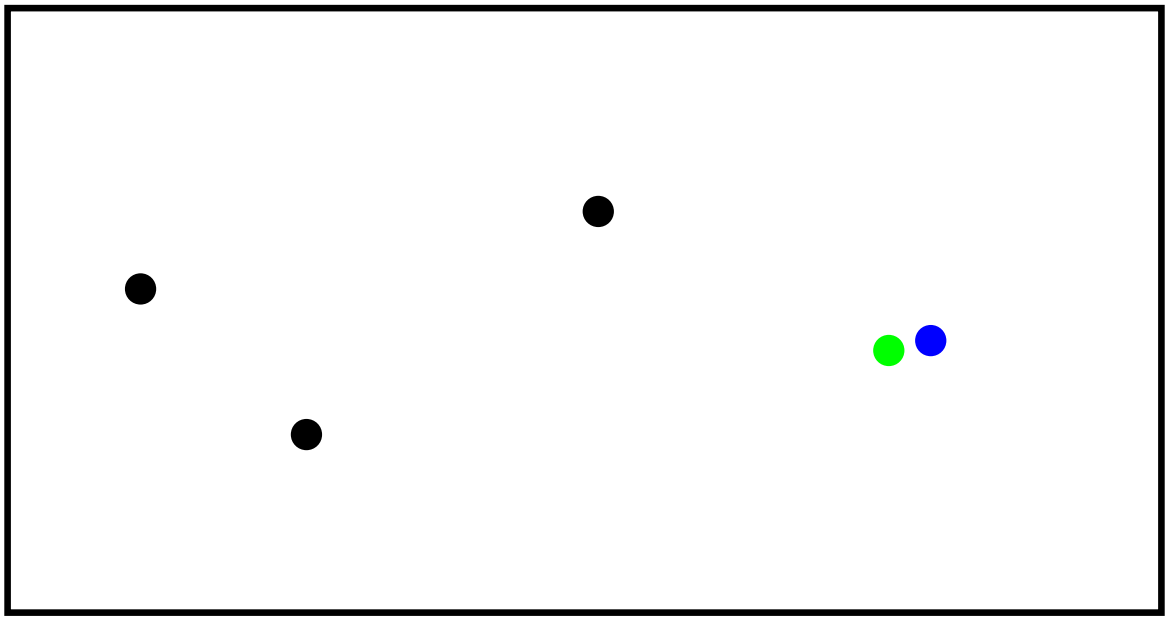
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

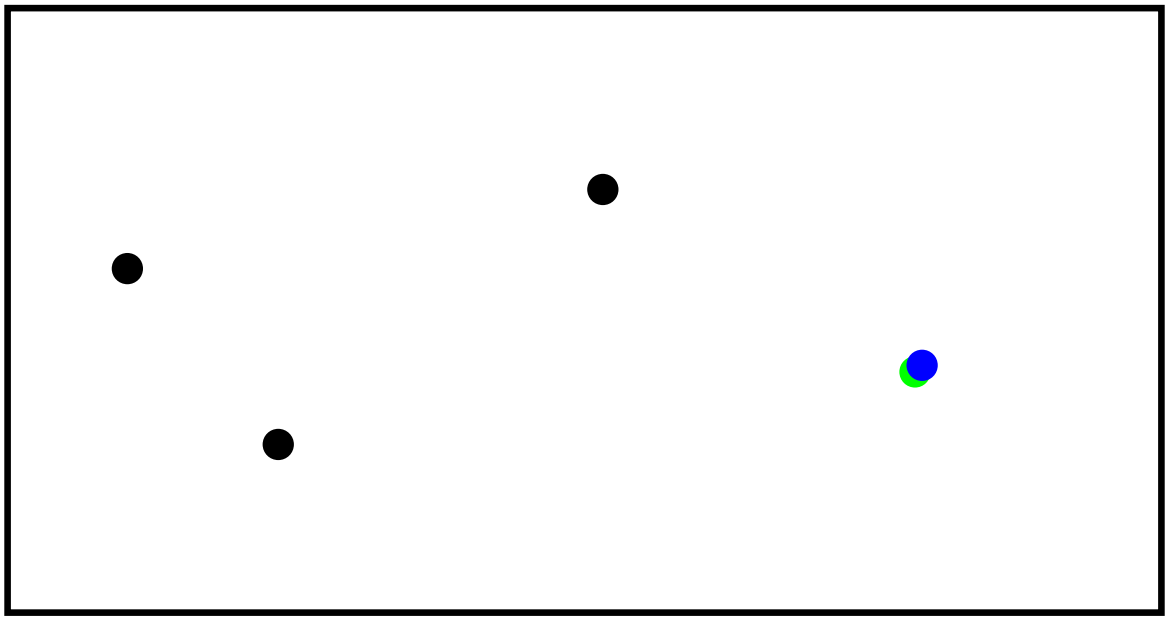
● ● → ● ●

● ● → ● ●

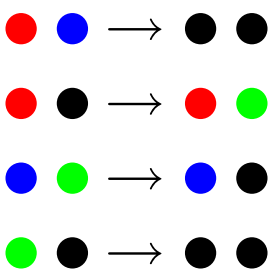
● ● → ● ●

● ● → ● ●

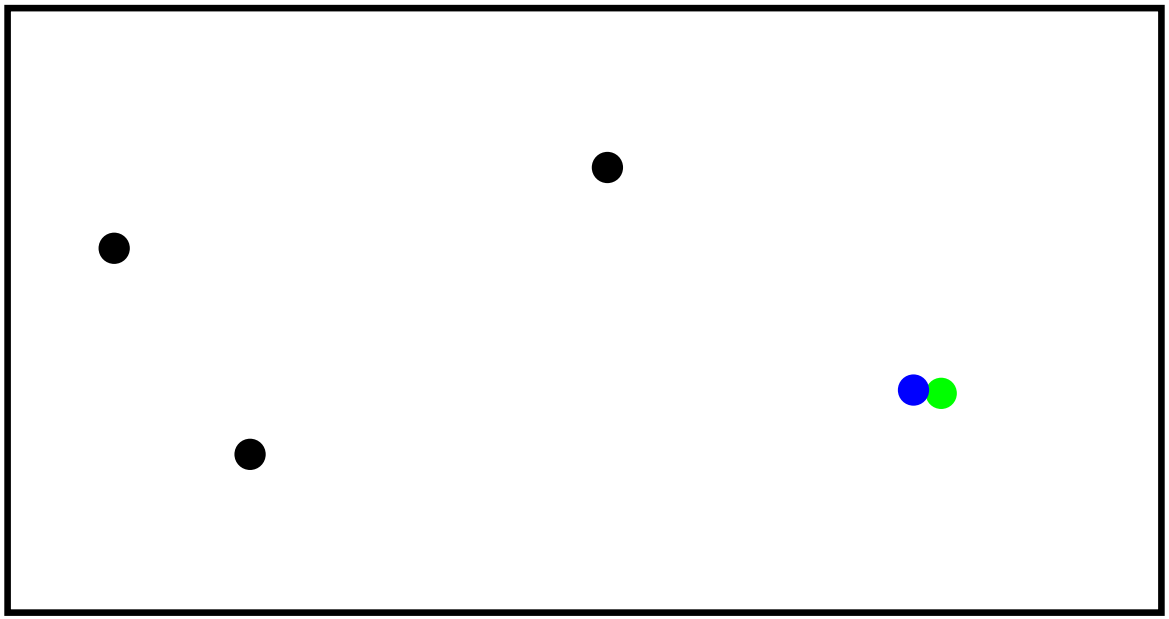
Exemple 1: partant de 3 ●, 2 ●



Program:



Exemple 1: partant de 3 ●, 2 ●



Program:

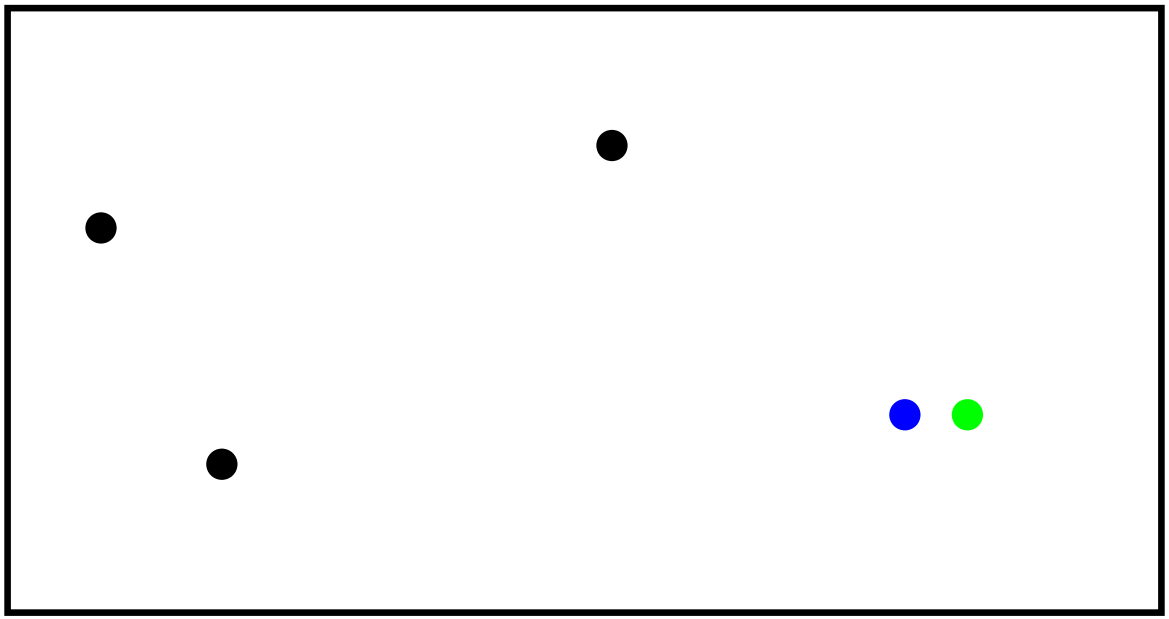
● ● → ● ●

● ● → ● ●

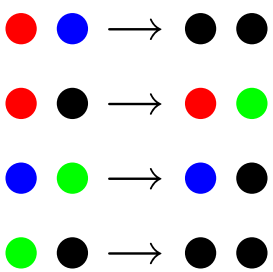
● ● → ● ●

● ● → ● ●

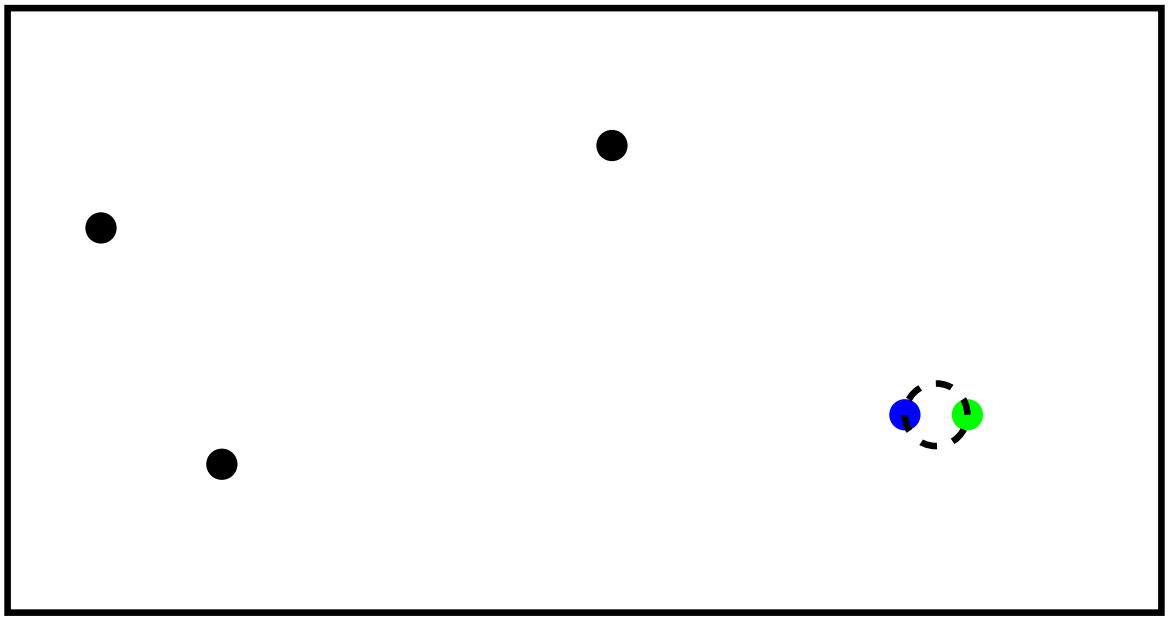
Exemple 1: partant de 3 ●, 2 ●



Program:



Exemple 1: partant de 3 ●, 2 ●



Program:

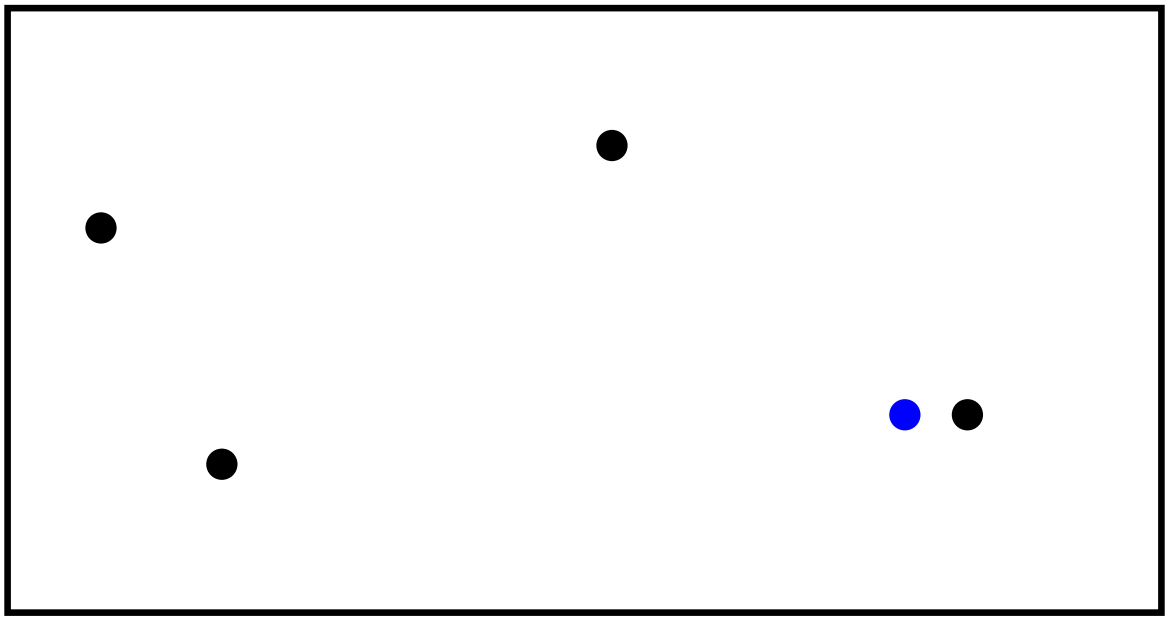
● ● → ● ●

● ● → ● ●

● ● → ● ●

● ● → ● ●

Exemple 1: partant de 3 ●, 2 ●



Program:

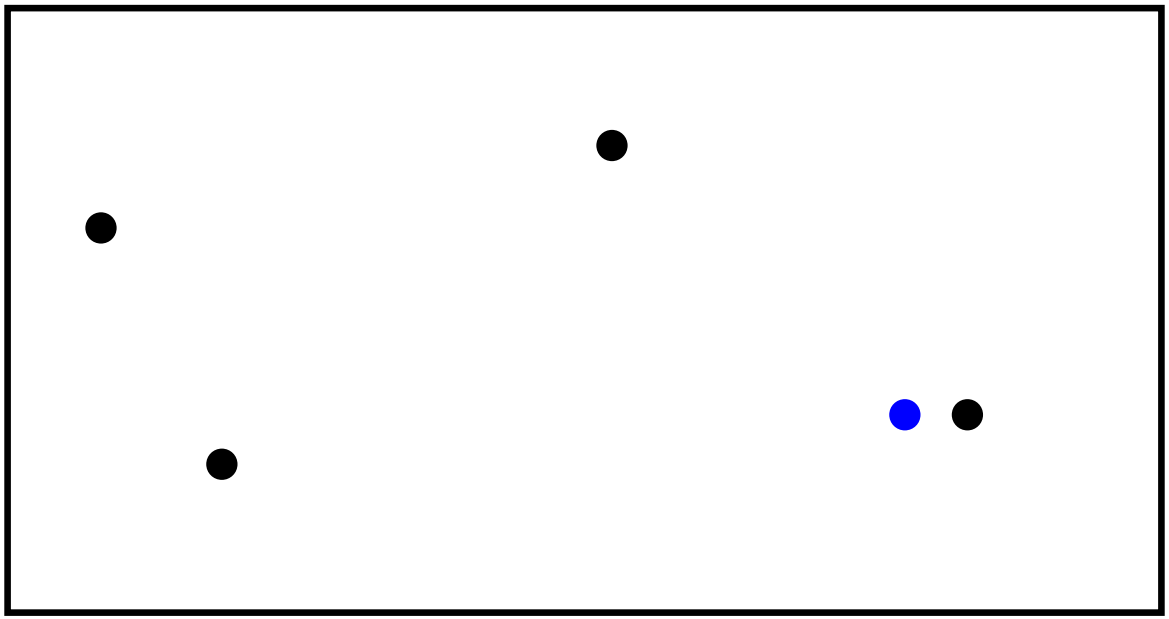
● ● → ● ●

● ● → ● ●

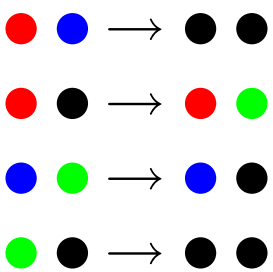
● ● → ● ●

● ● → ● ●

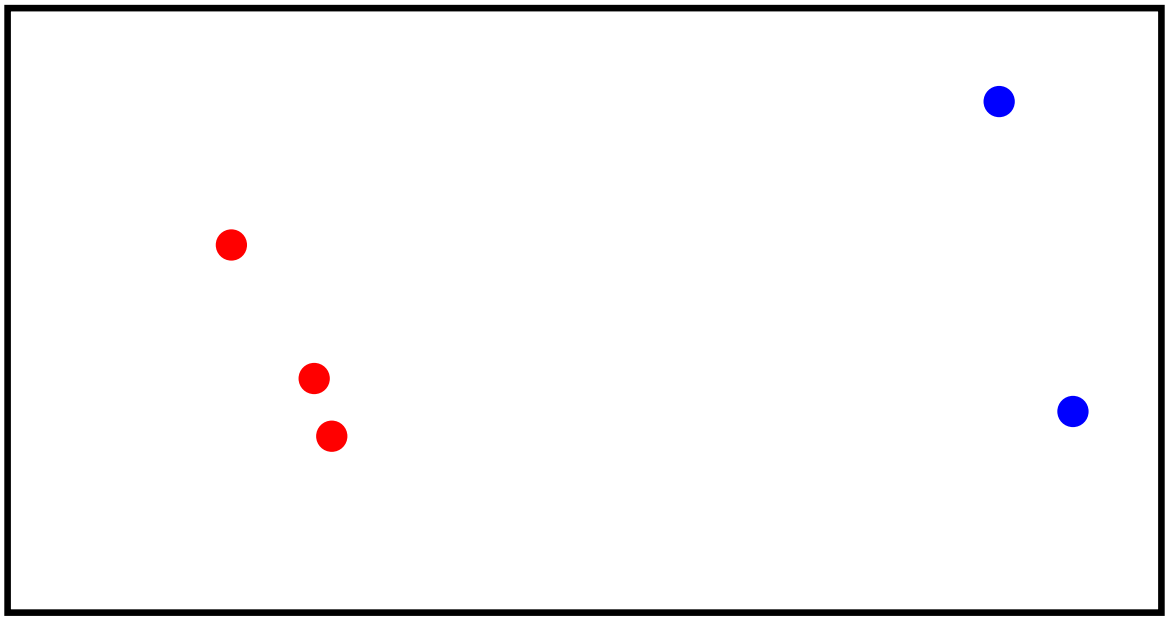
Example 1: Résultat final



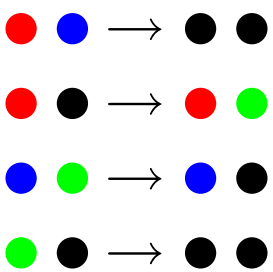
Program:



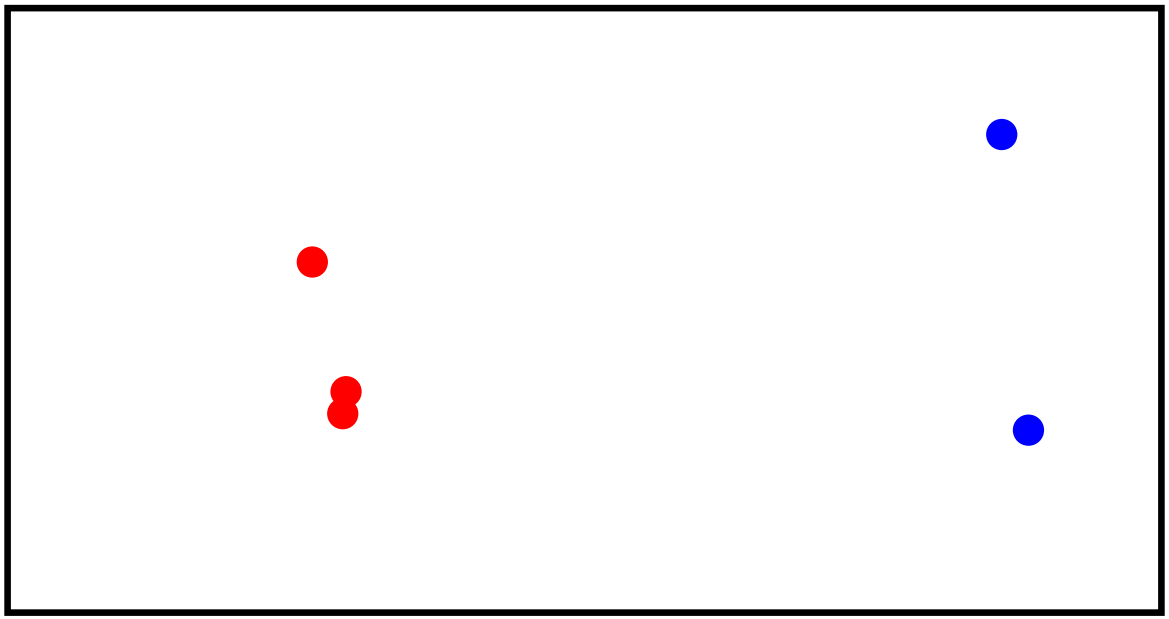
Exemple 2: partant de 2 ●, 3●



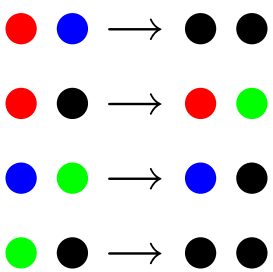
Program:



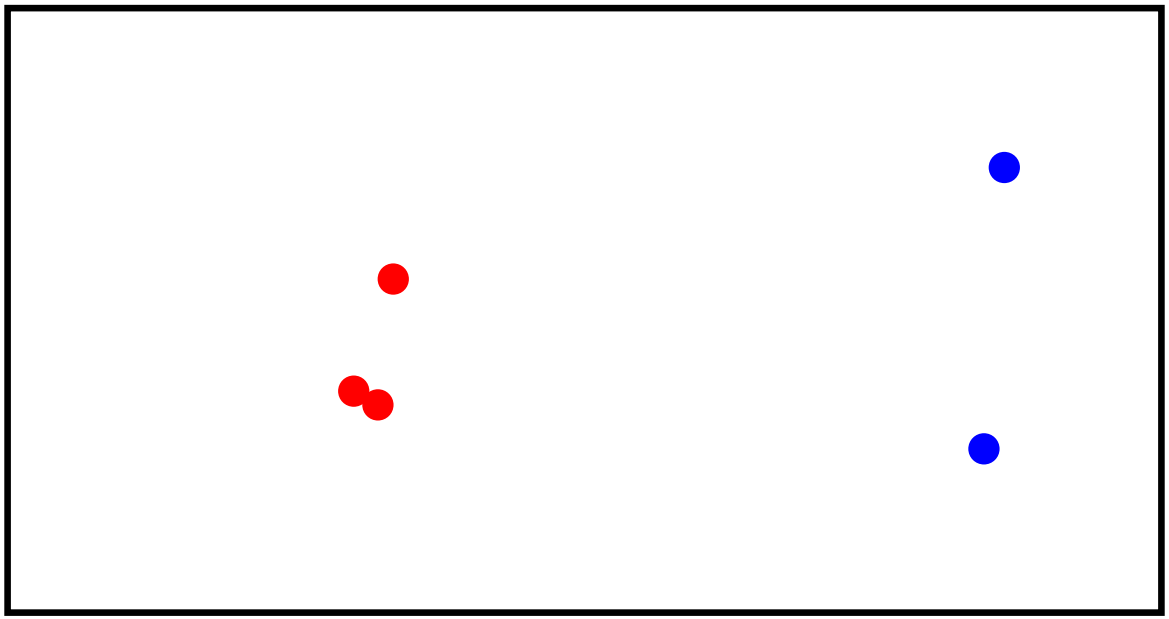
Exemple 2: partant de 2 ●, 3●



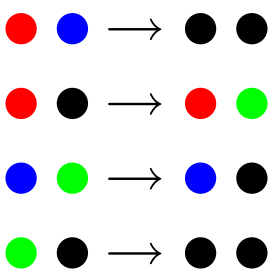
Program:



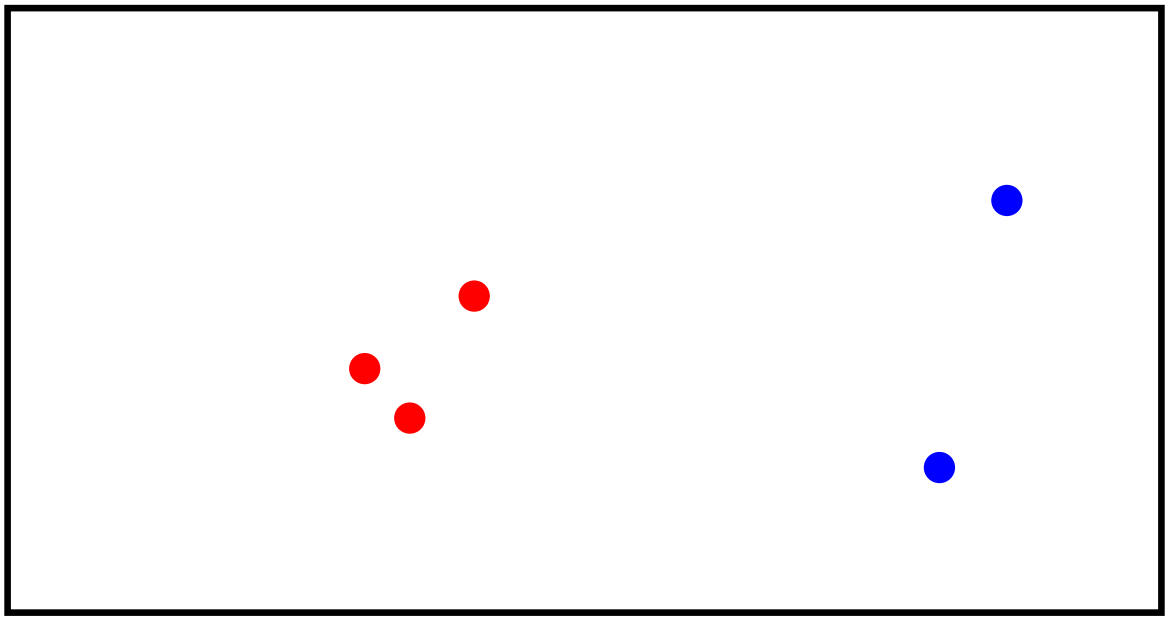
Exemple 2: partant de 2 ●, 3●



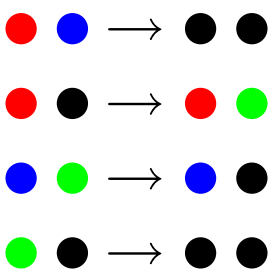
Program:



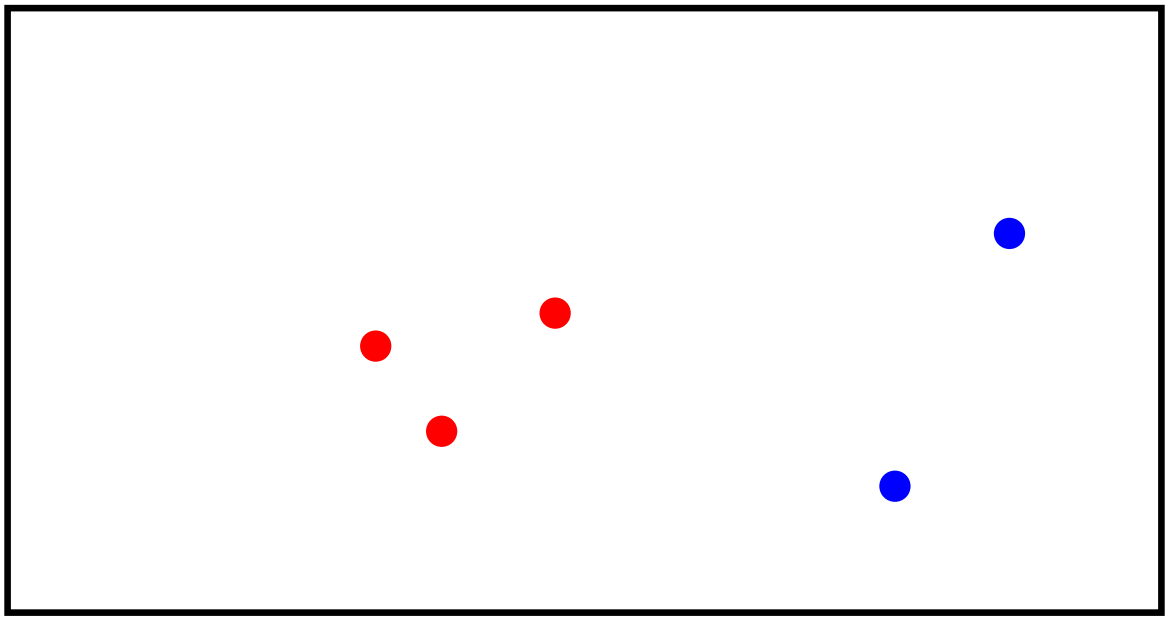
Exemple 2: partant de 2 ●, 3●



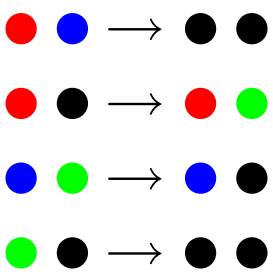
Program:



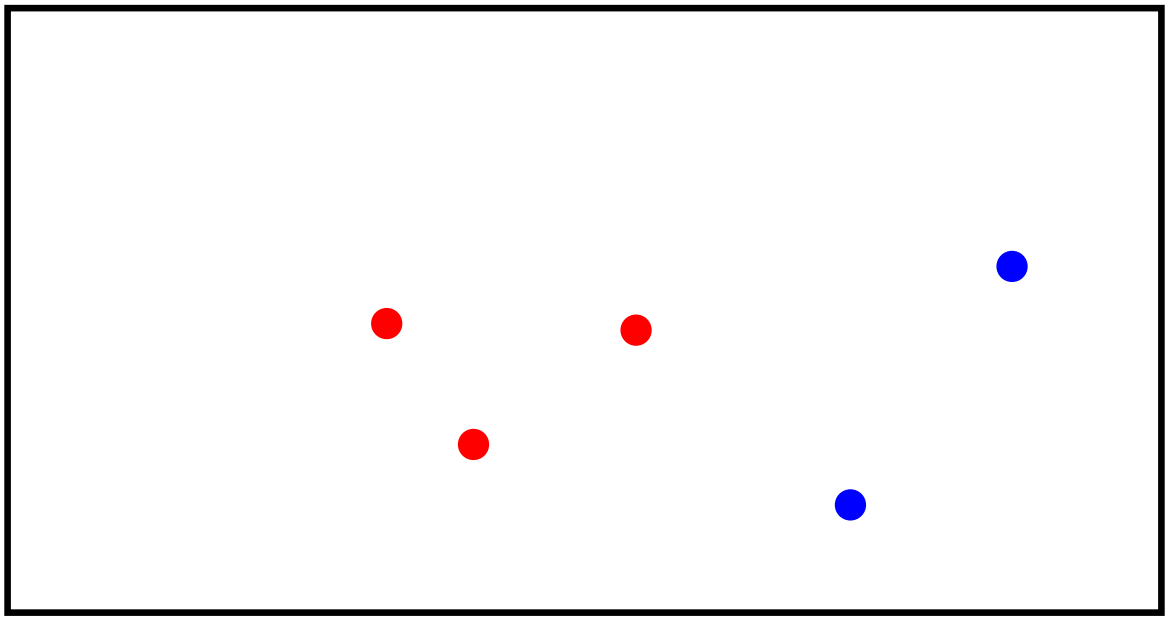
Exemple 2: partant de 2 ●, 3●



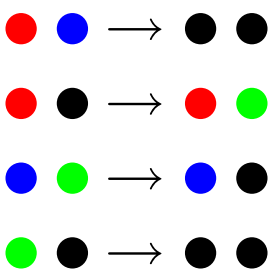
Program:



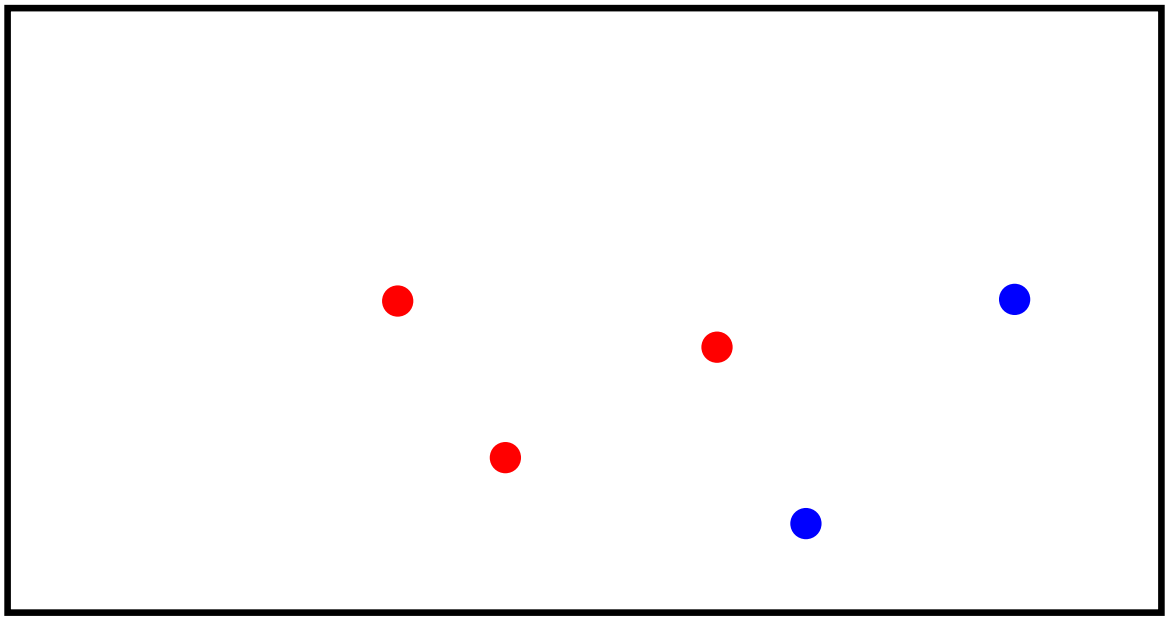
Exemple 2: partant de 2 ●, 3●



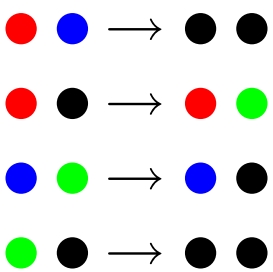
Program:



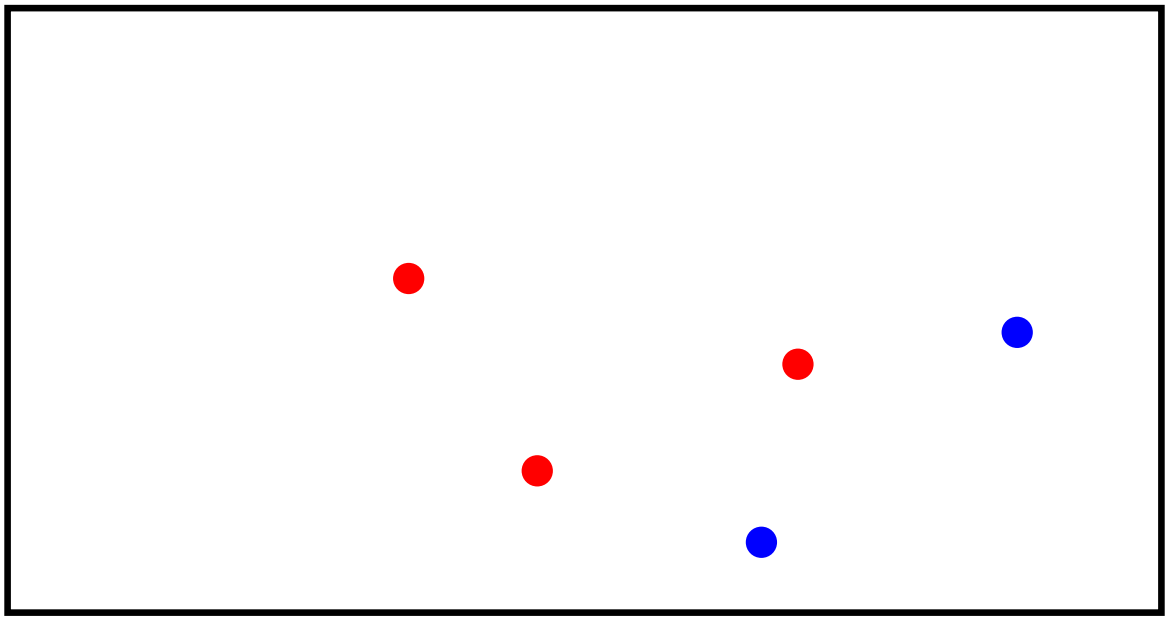
Exemple 2: partant de 2 ●, 3●



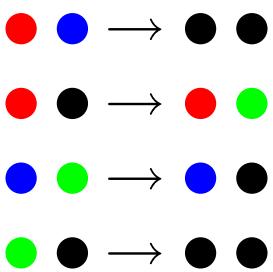
Program:



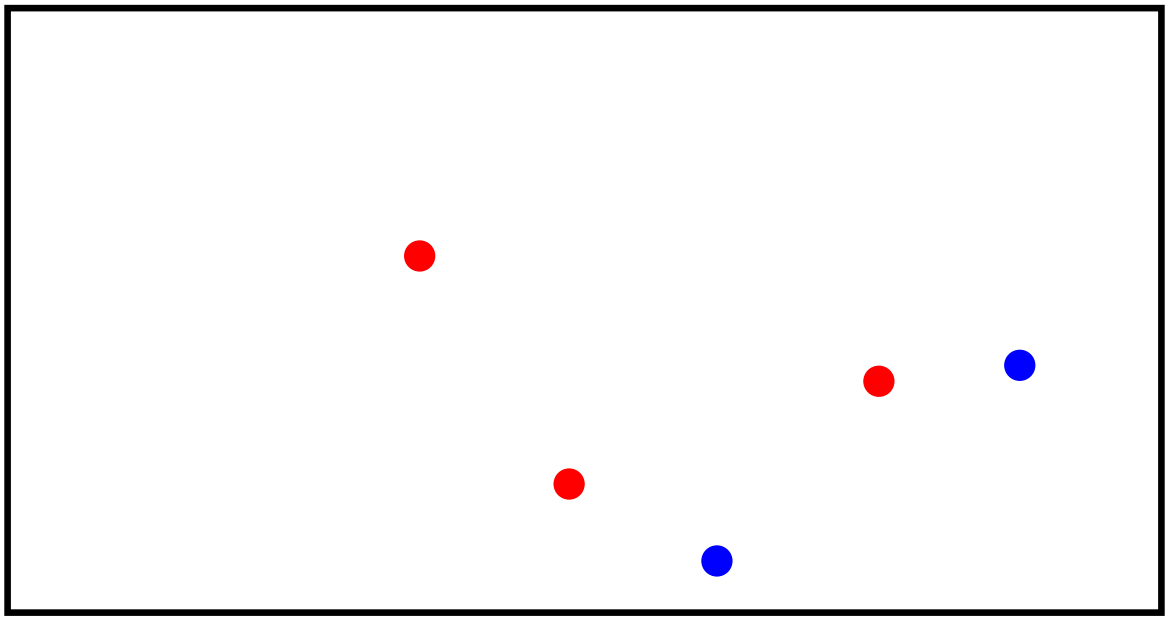
Exemple 2: partant de 2 ●, 3●



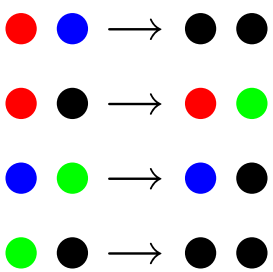
Program:



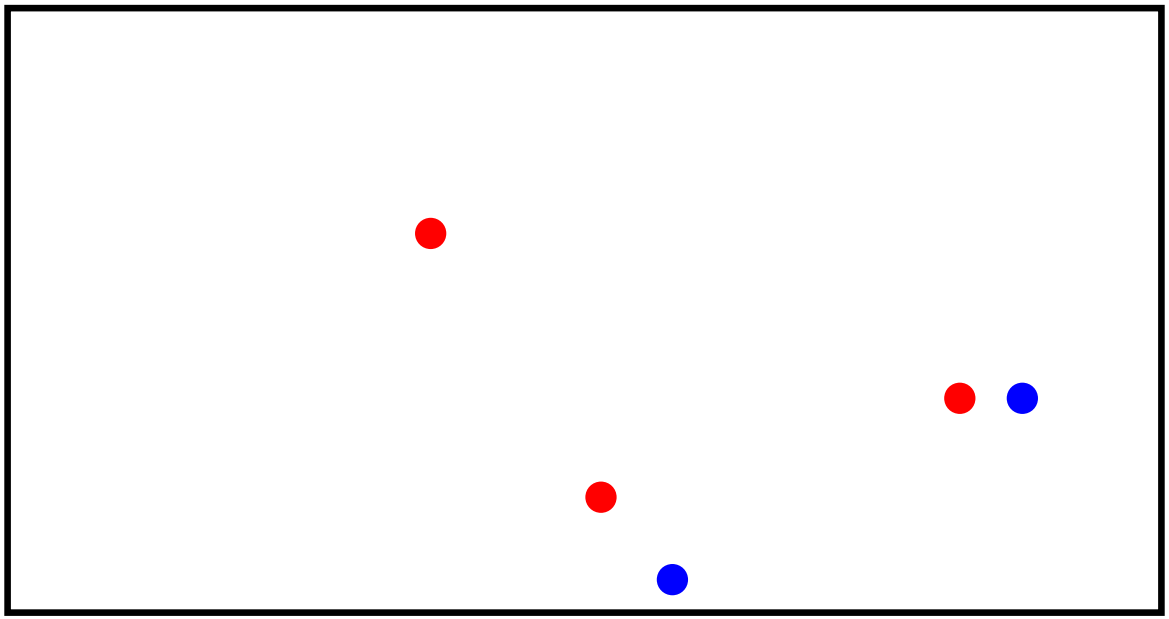
Exemple 2: partant de 2 ●, 3●



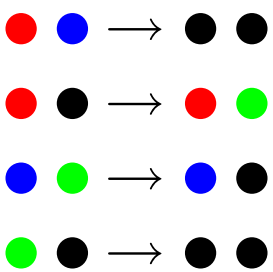
Program:



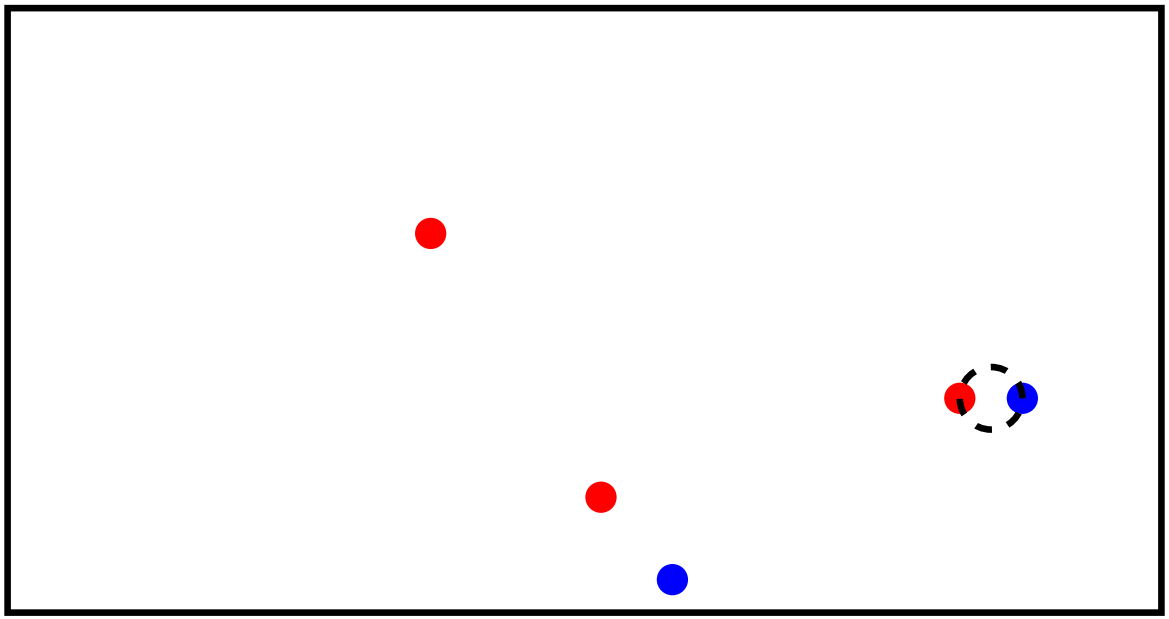
Exemple 2: partant de 2 ●, 3●



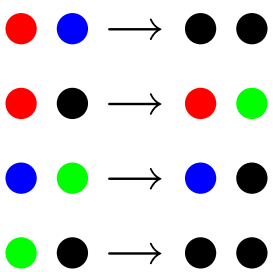
Program:



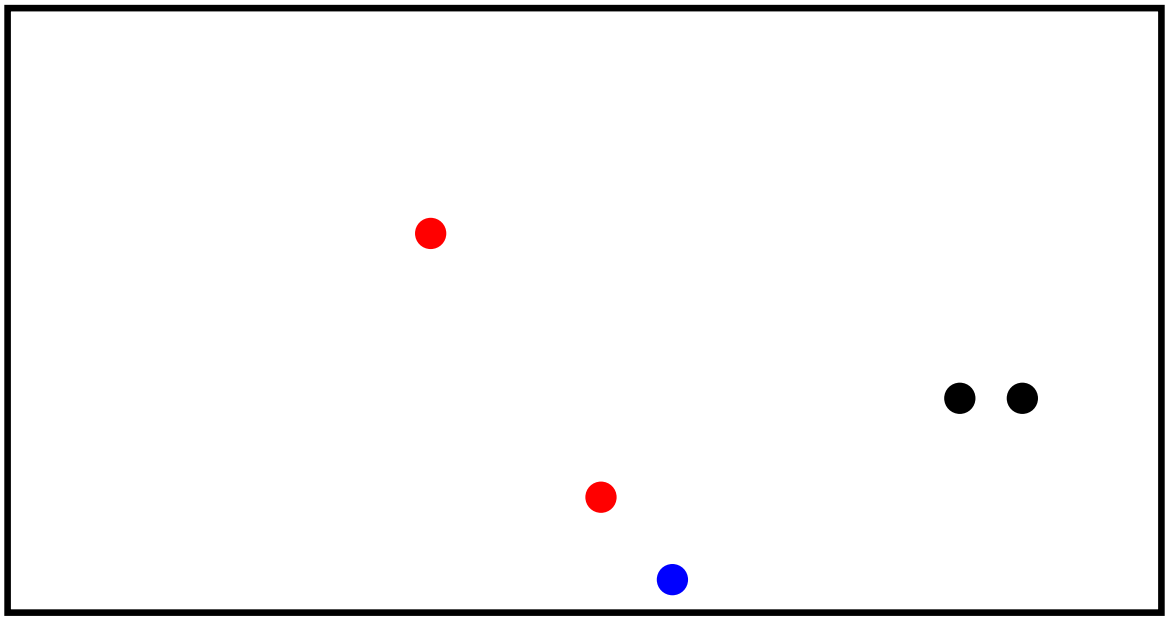
Exemple 2: partant de 2 ●, 3●



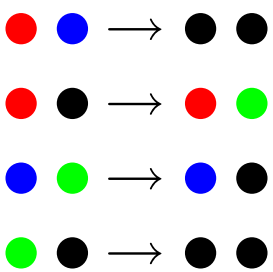
Program:



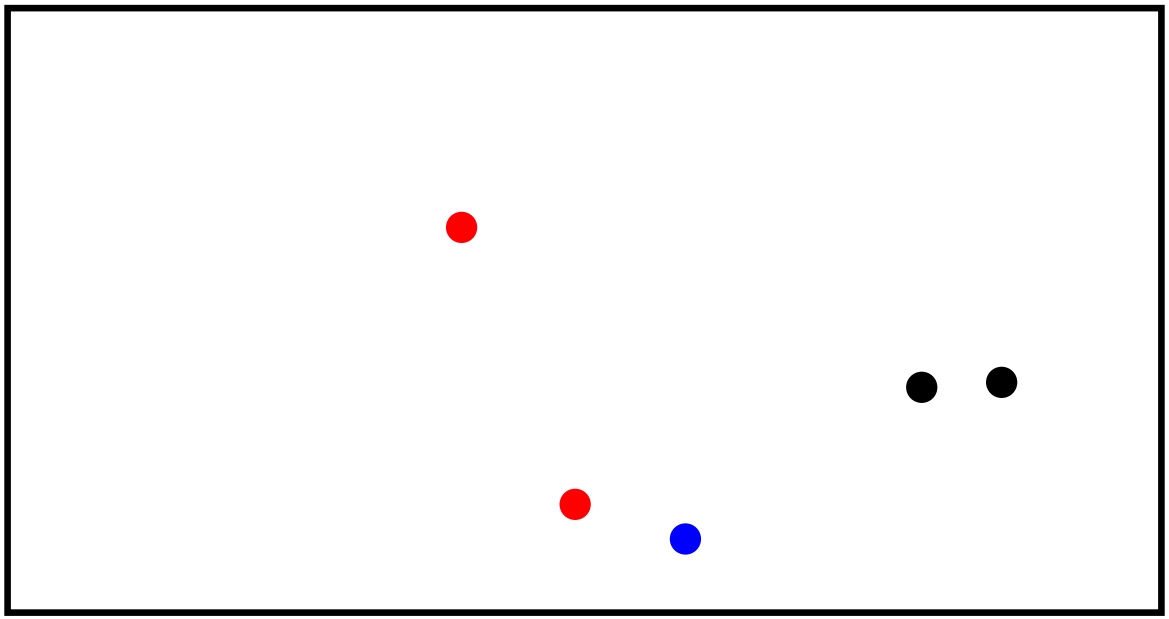
Exemple 2: partant de 2 ●, 3●



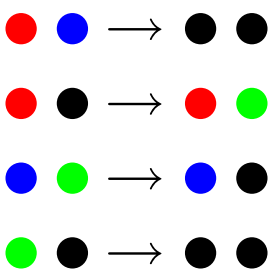
Program:



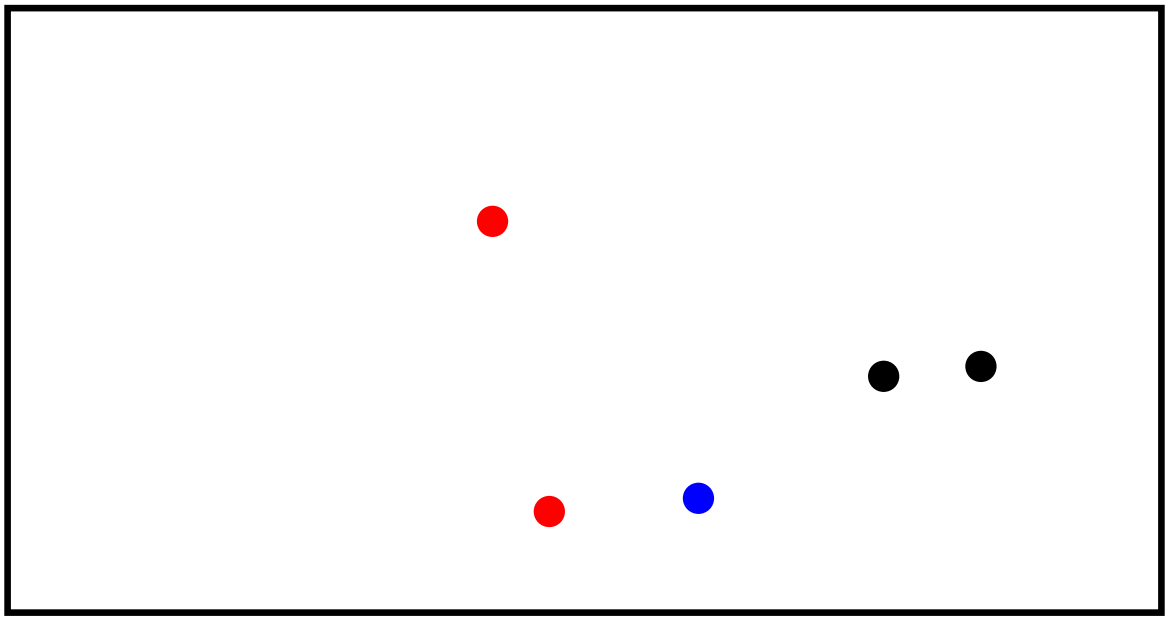
Exemple 2: partant de 2 ●, 3●



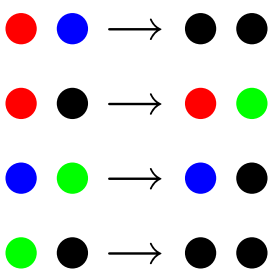
Program:



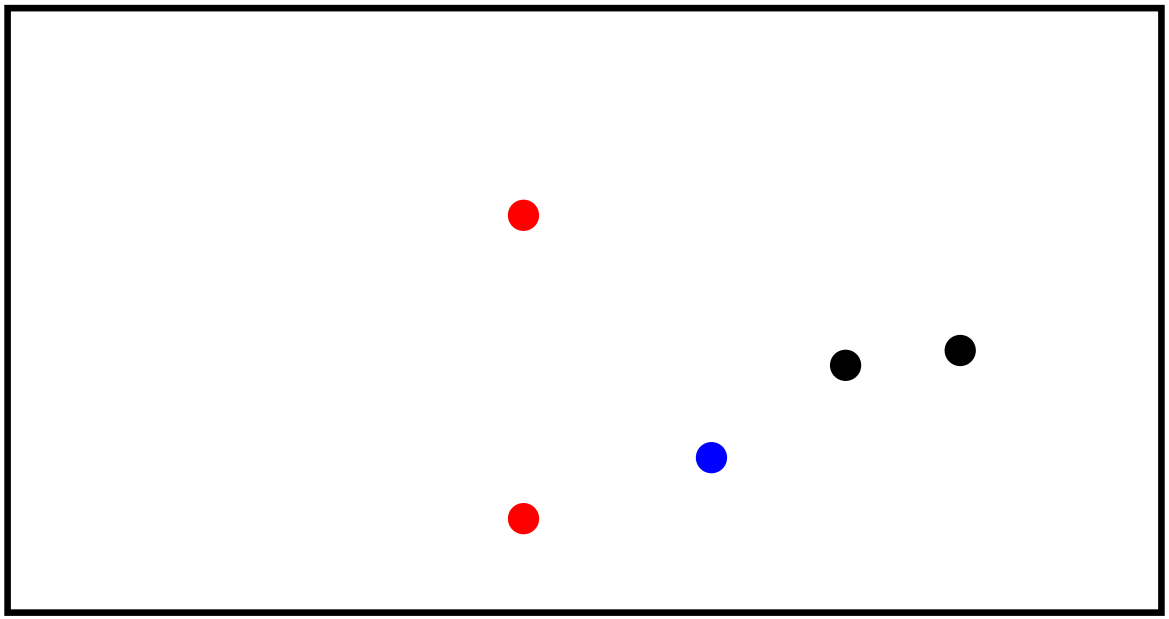
Exemple 2: partant de 2 ●, 3●



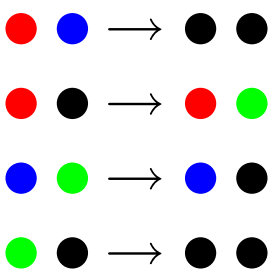
Program:



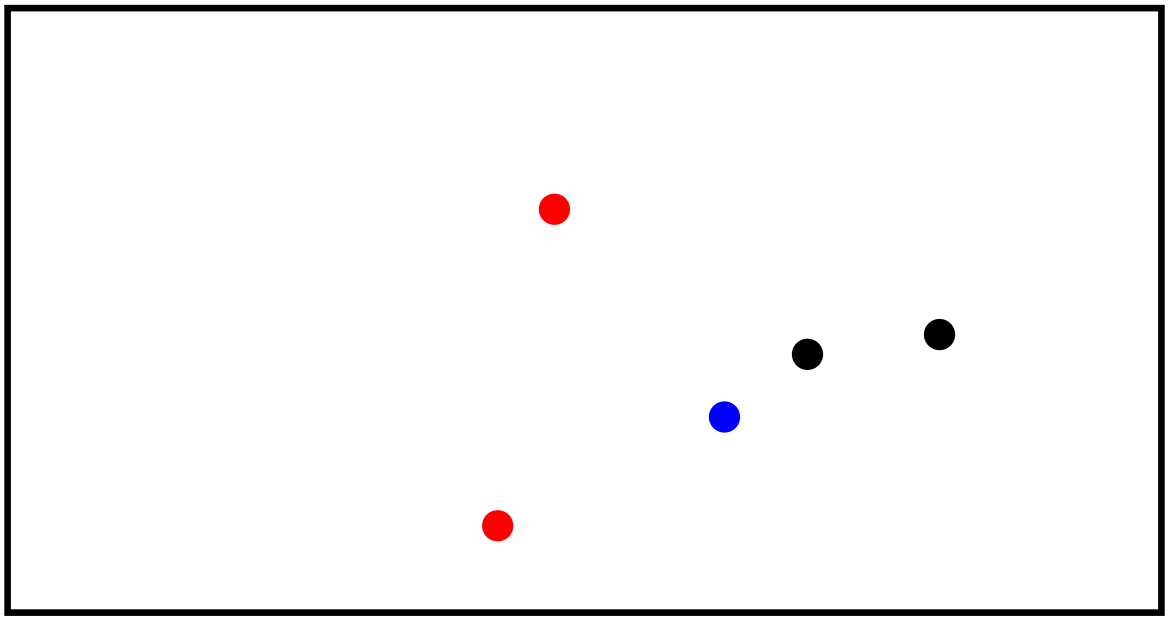
Exemple 2: partant de 2 ●, 3●



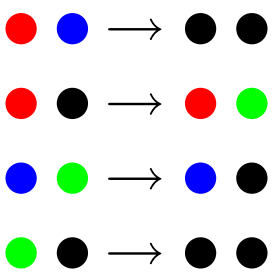
Program:



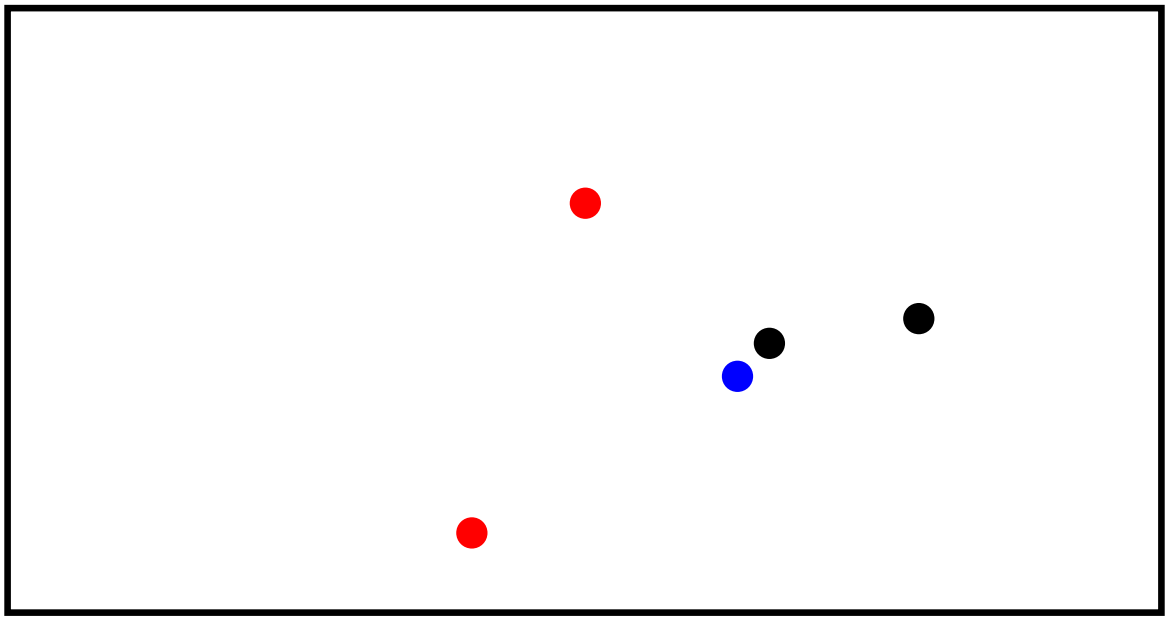
Exemple 2: partant de 2 ●, 3●



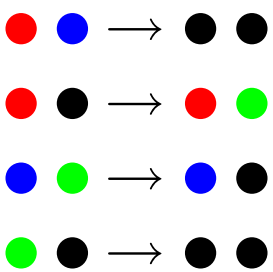
Program:



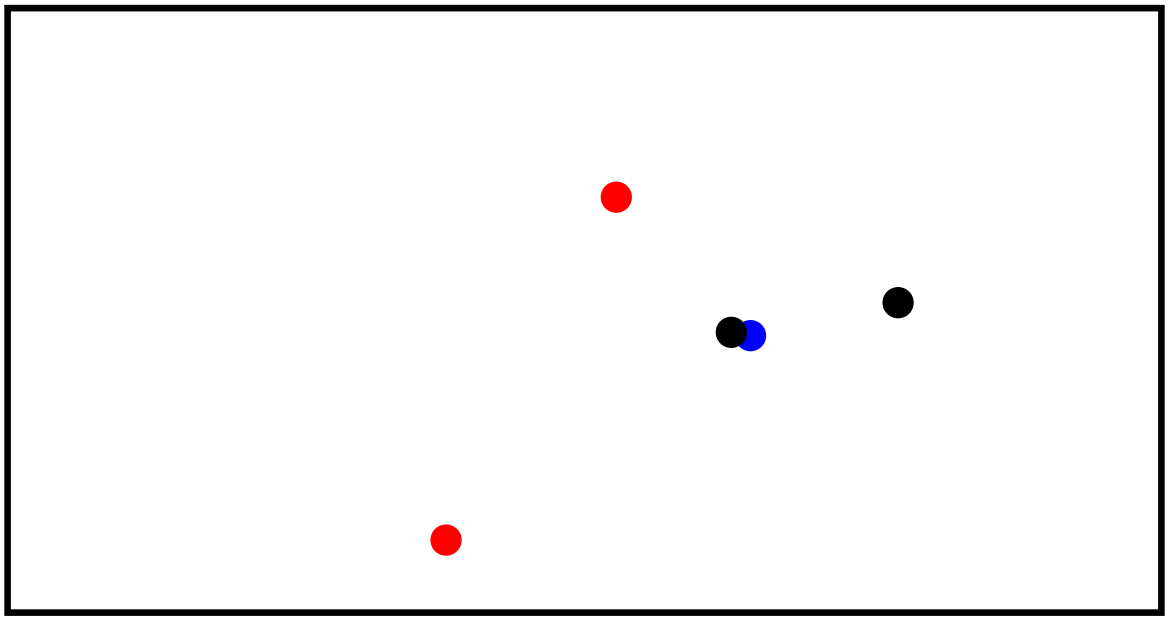
Exemple 2: partant de 2 ●, 3●



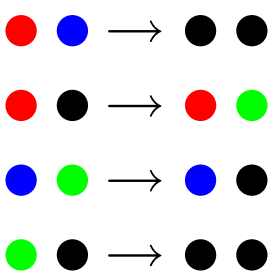
Program:



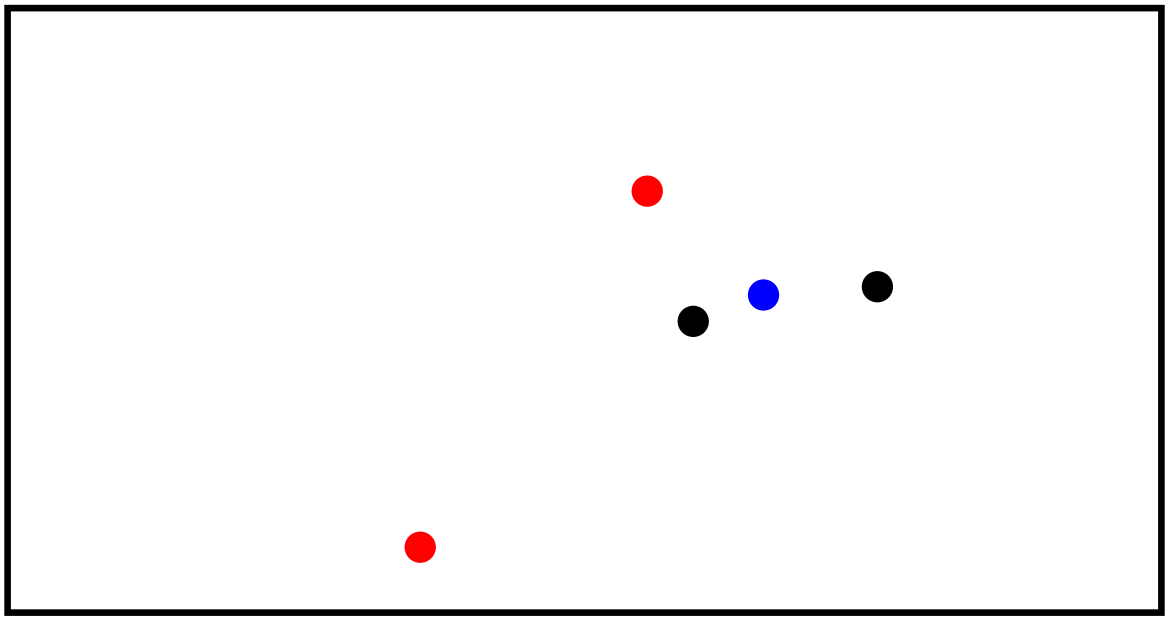
Exemple 2: partant de 2 ●, 3●



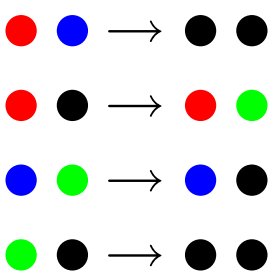
Program:



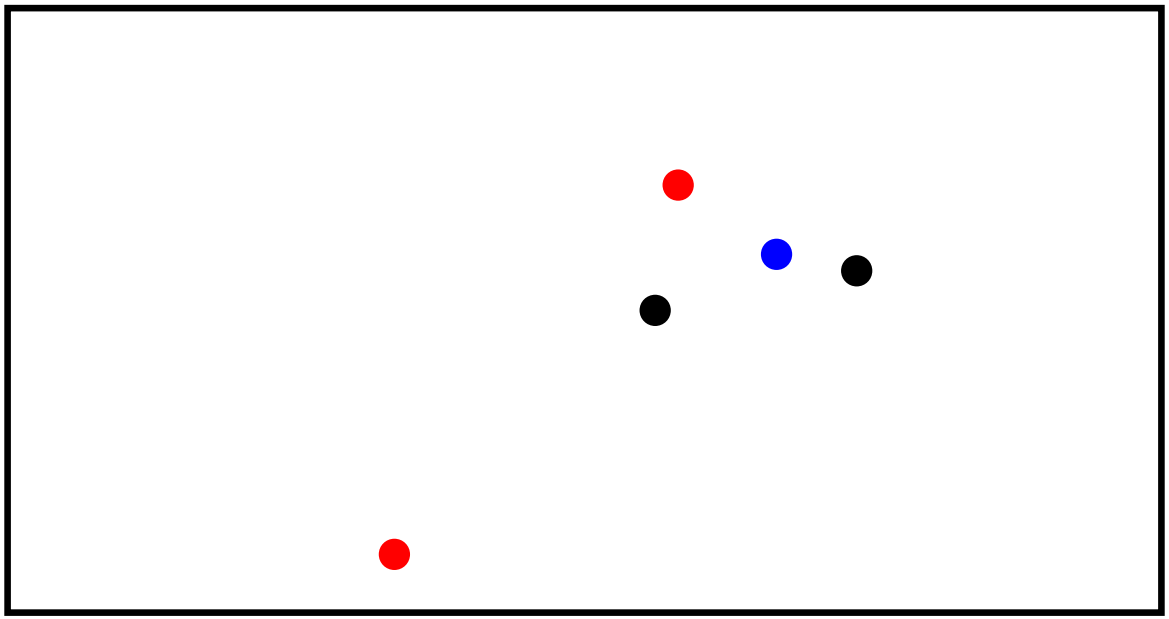
Exemple 2: partant de 2 ●, 3●



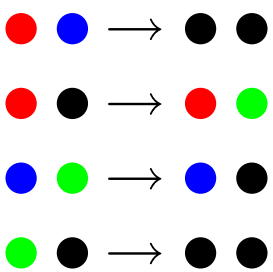
Program:



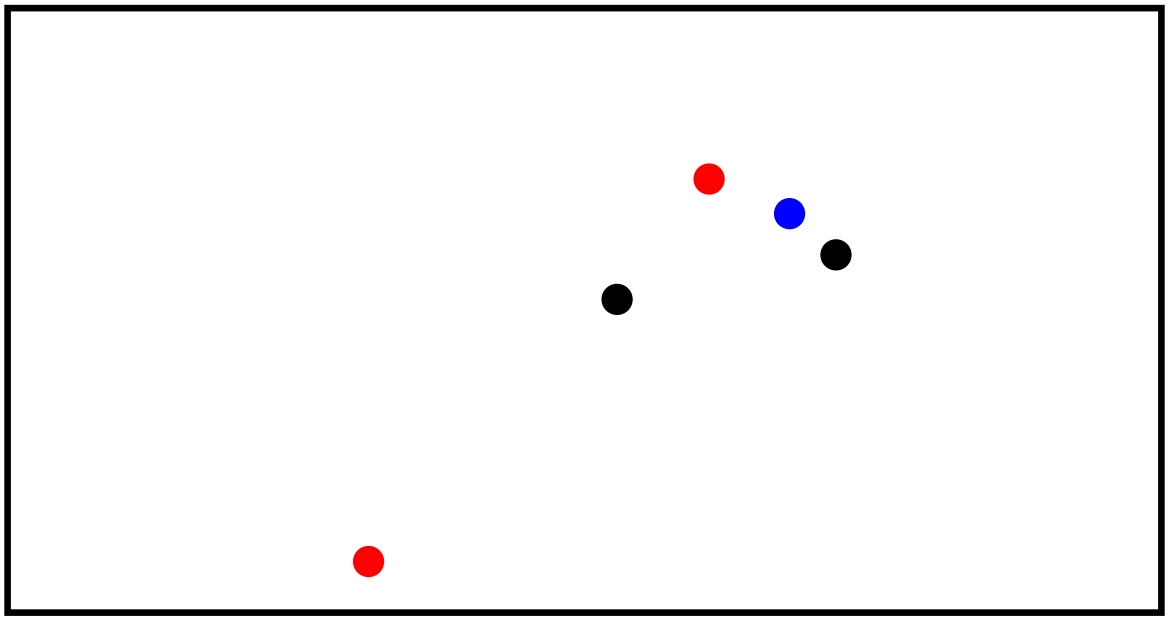
Exemple 2: partant de 2 ●, 3●



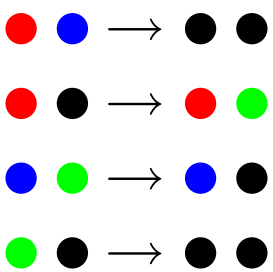
Program:



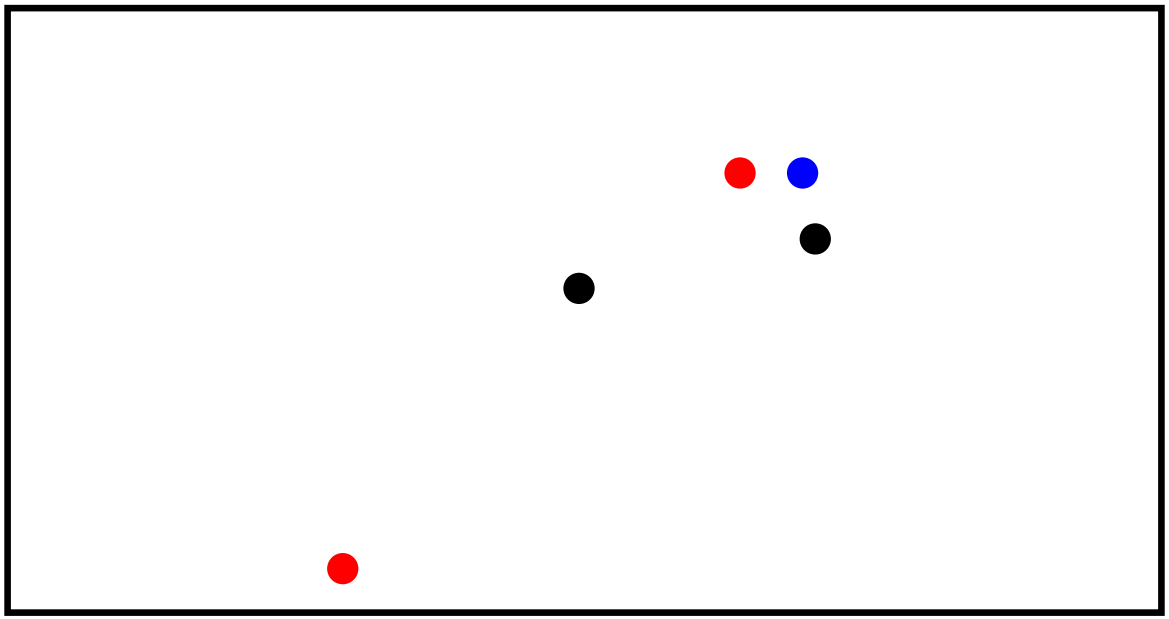
Exemple 2: partant de 2 ●, 3●



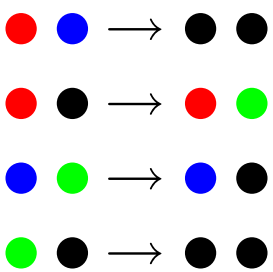
Program:



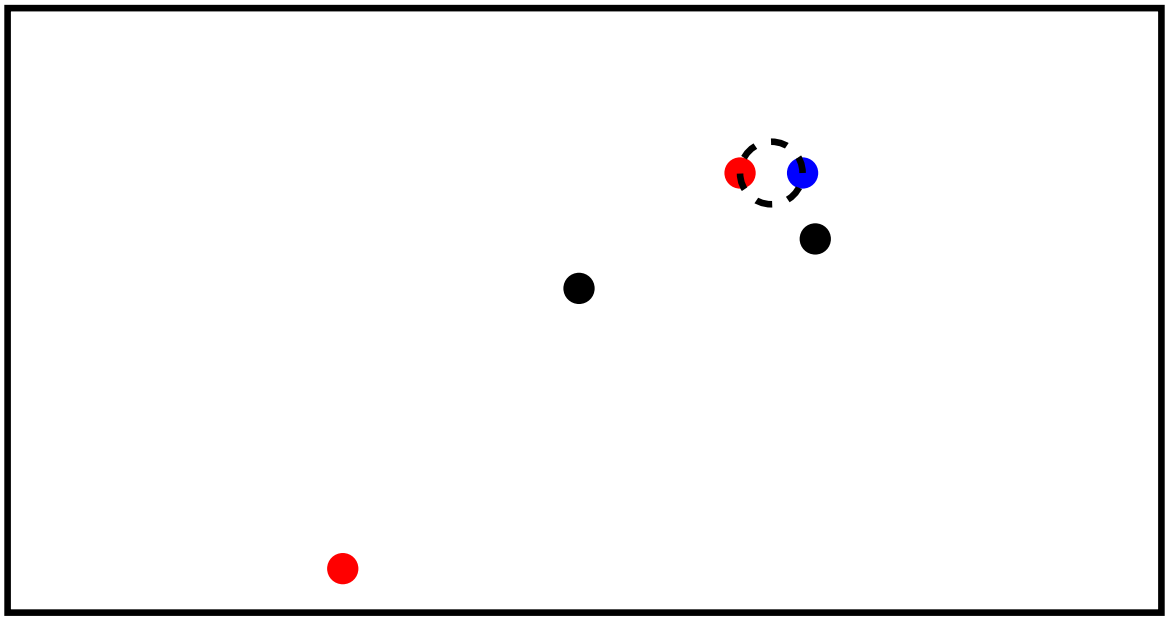
Exemple 2: partant de 2 ●, 3●



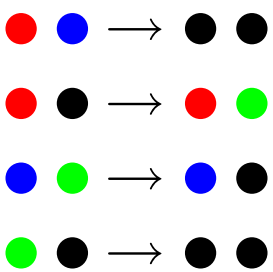
Program:



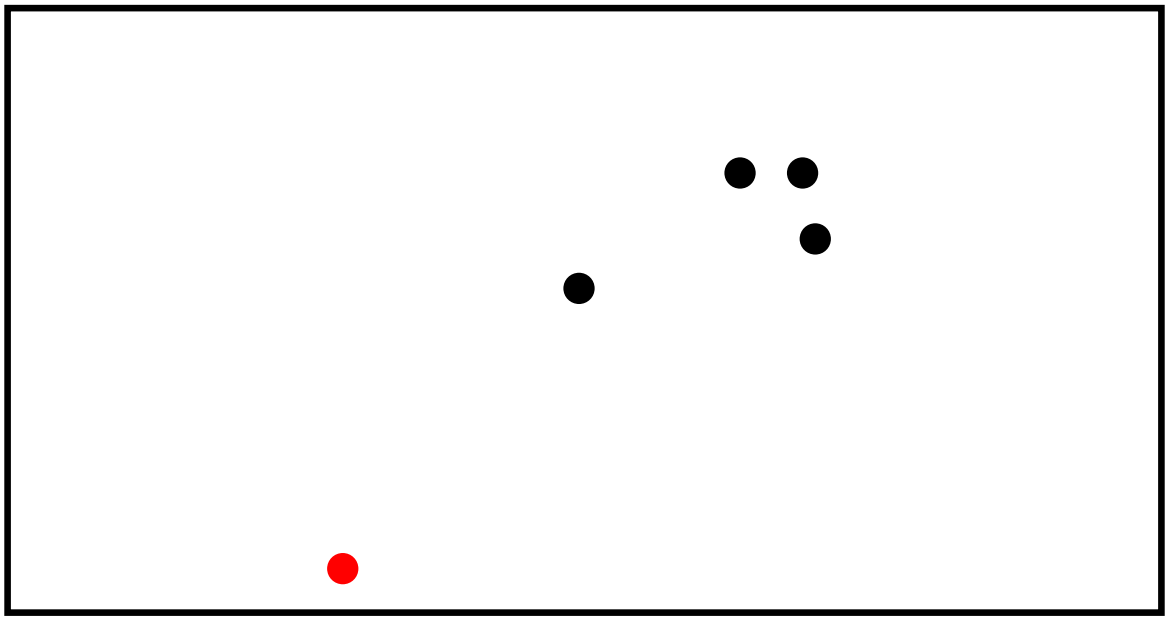
Exemple 2: partant de 2 ●, 3●



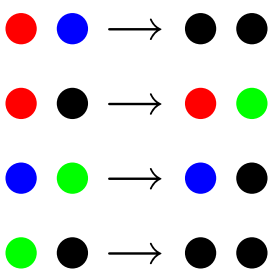
Program:



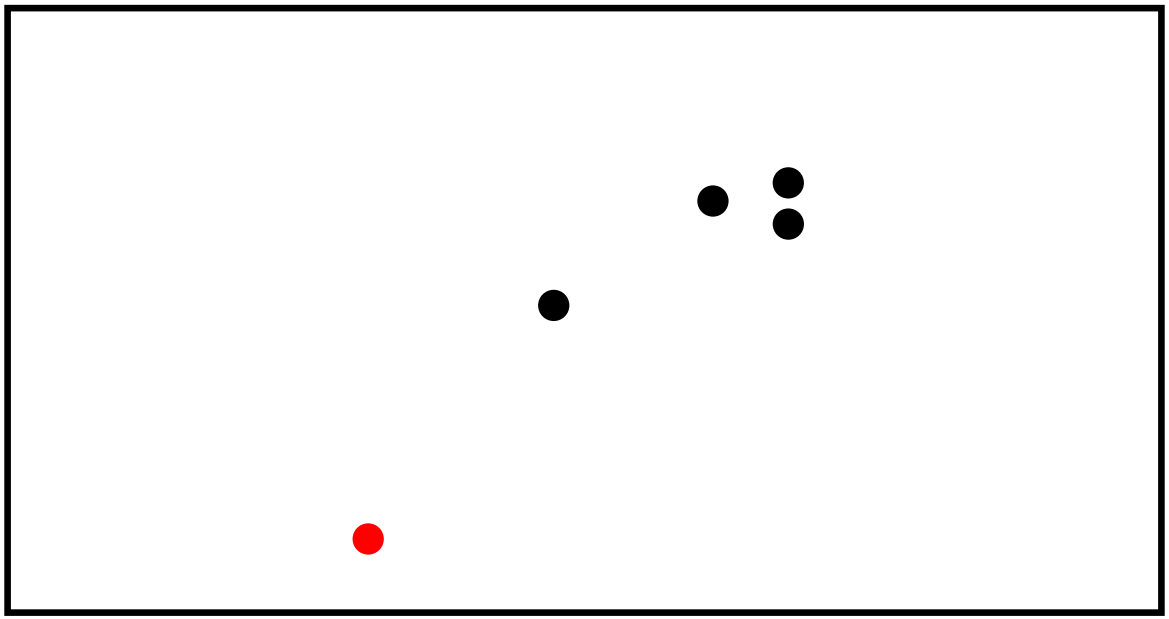
Exemple 2: partant de 2 ●, 3●



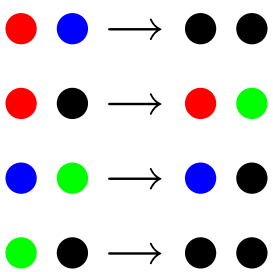
Program:



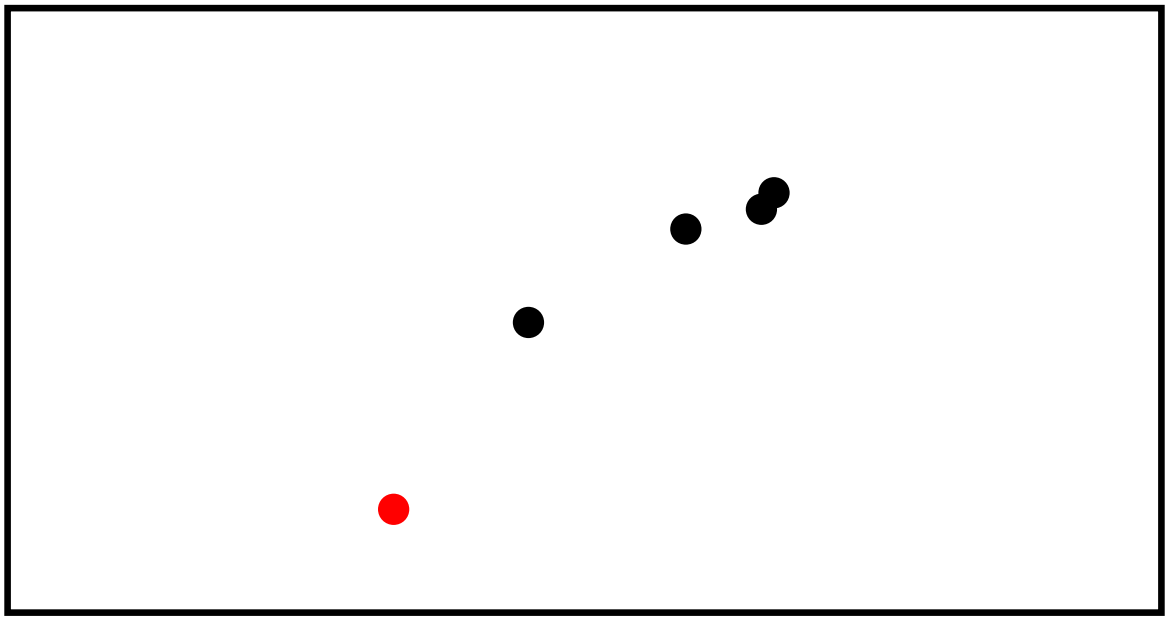
Exemple 2: partant de 2 ●, 3●



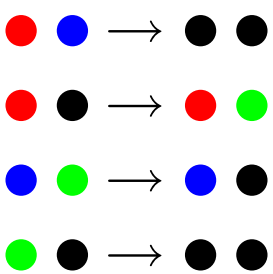
Program:



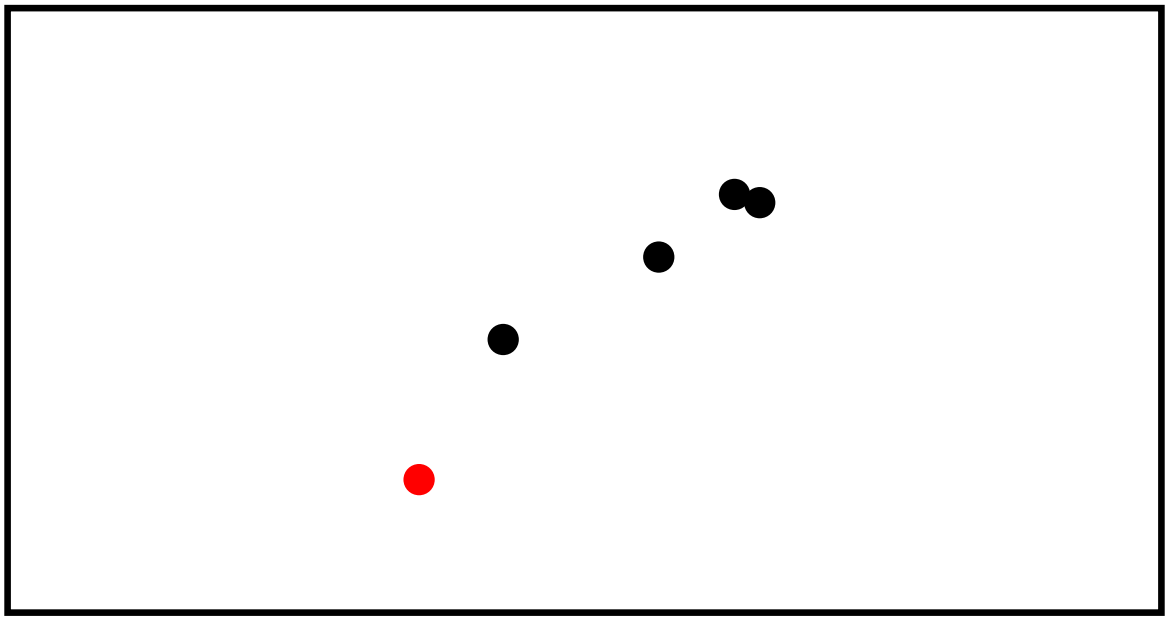
Exemple 2: partant de 2 ●, 3●



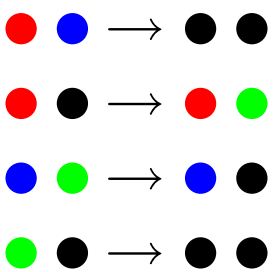
Program:



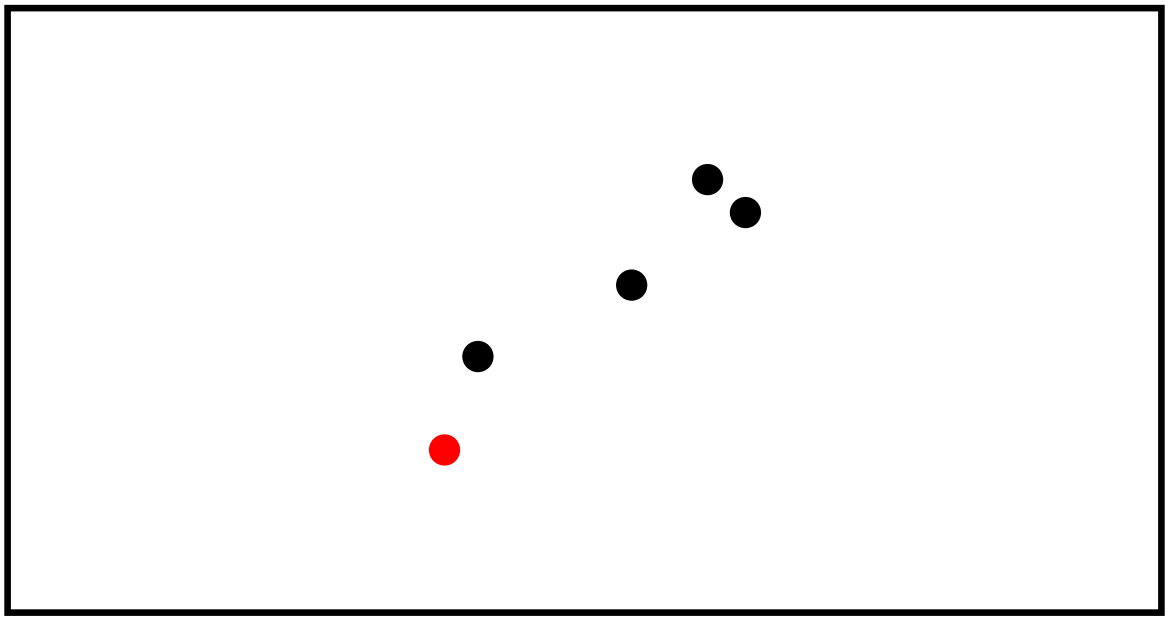
Exemple 2: partant de 2 ●, 3●



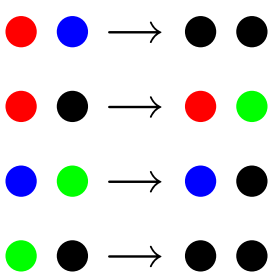
Program:



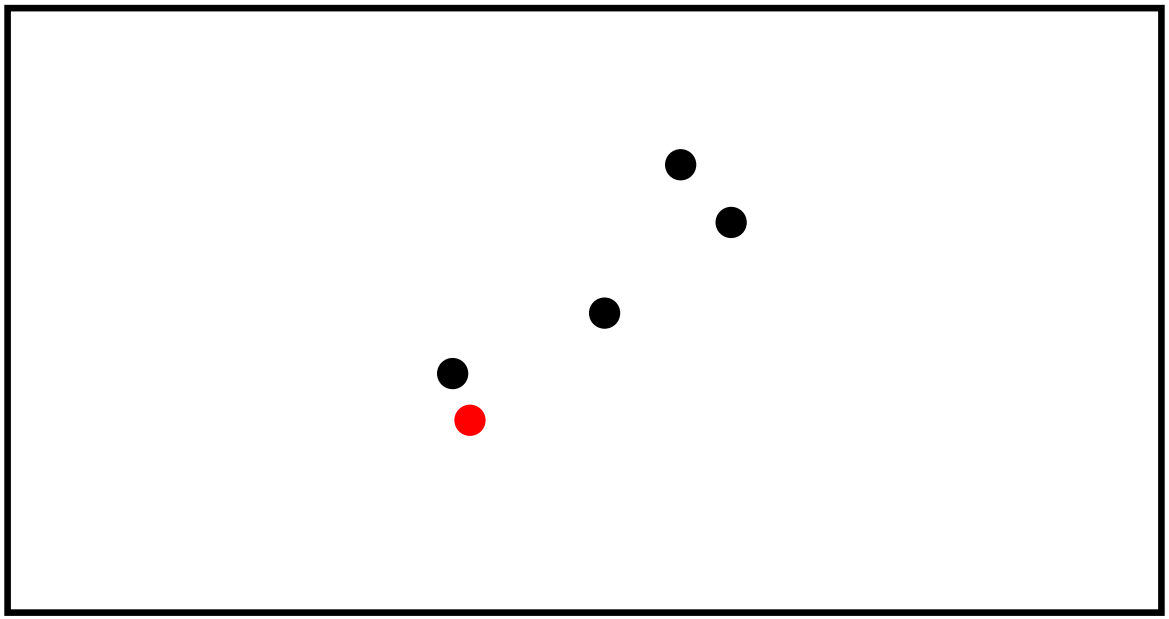
Exemple 2: partant de 2 ●, 3●



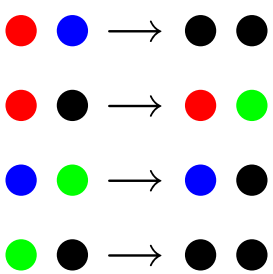
Program:



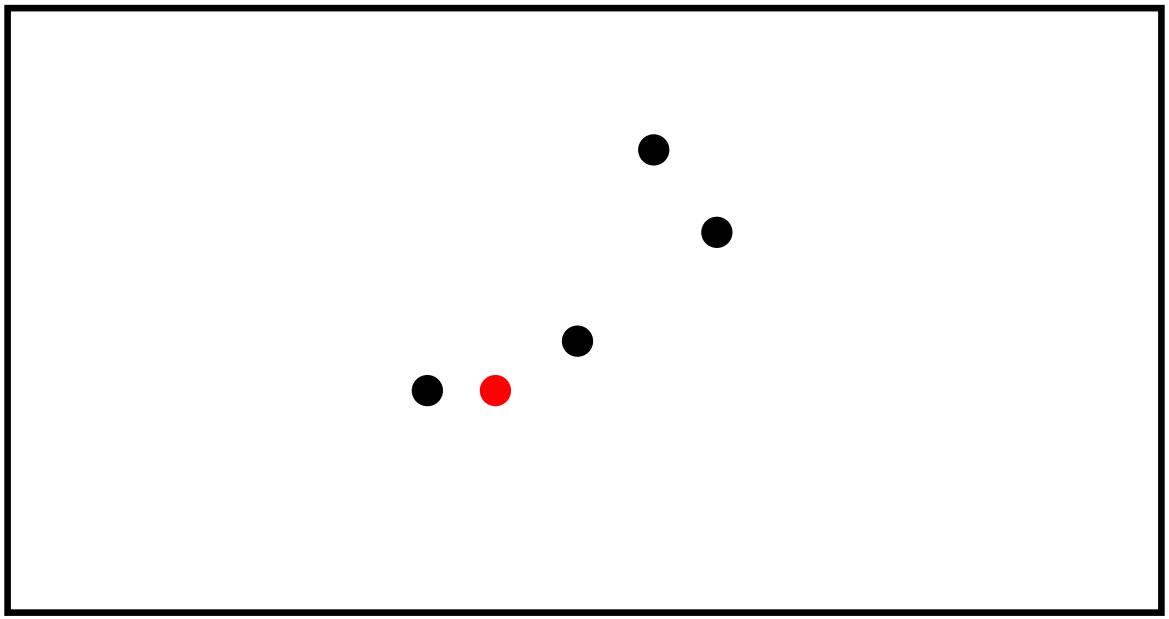
Exemple 2: partant de 2 ●, 3●



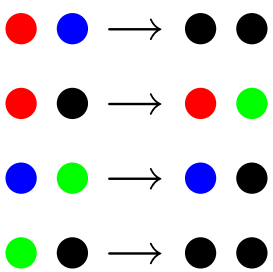
Program:



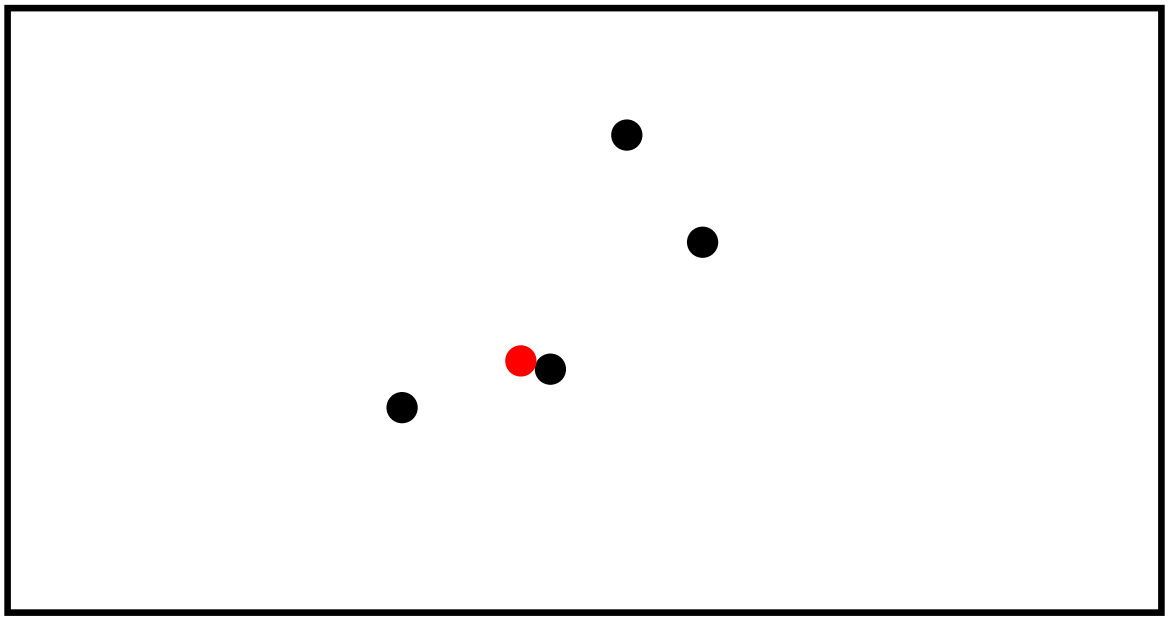
Exemple 2: partant de 2 ●, 3●



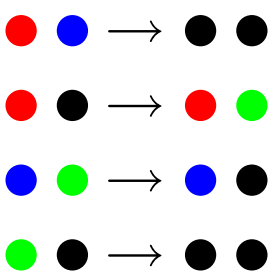
Program:



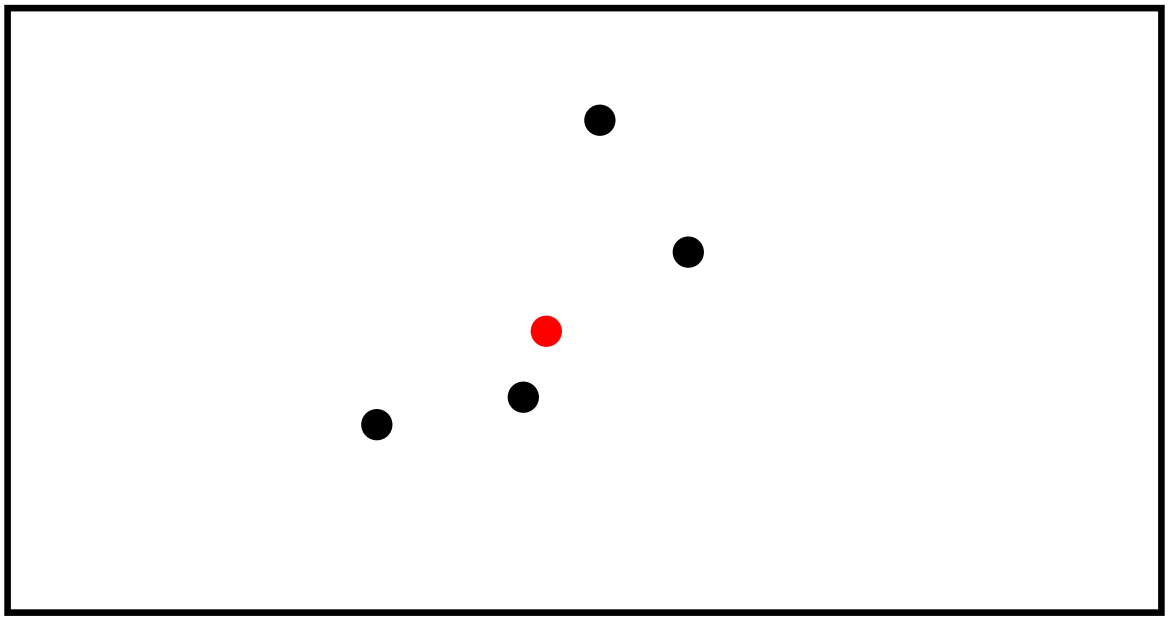
Exemple 2: partant de 2 ●, 3●



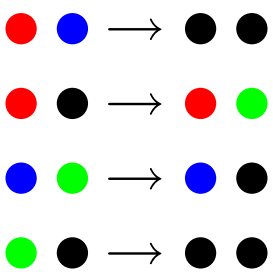
Program:



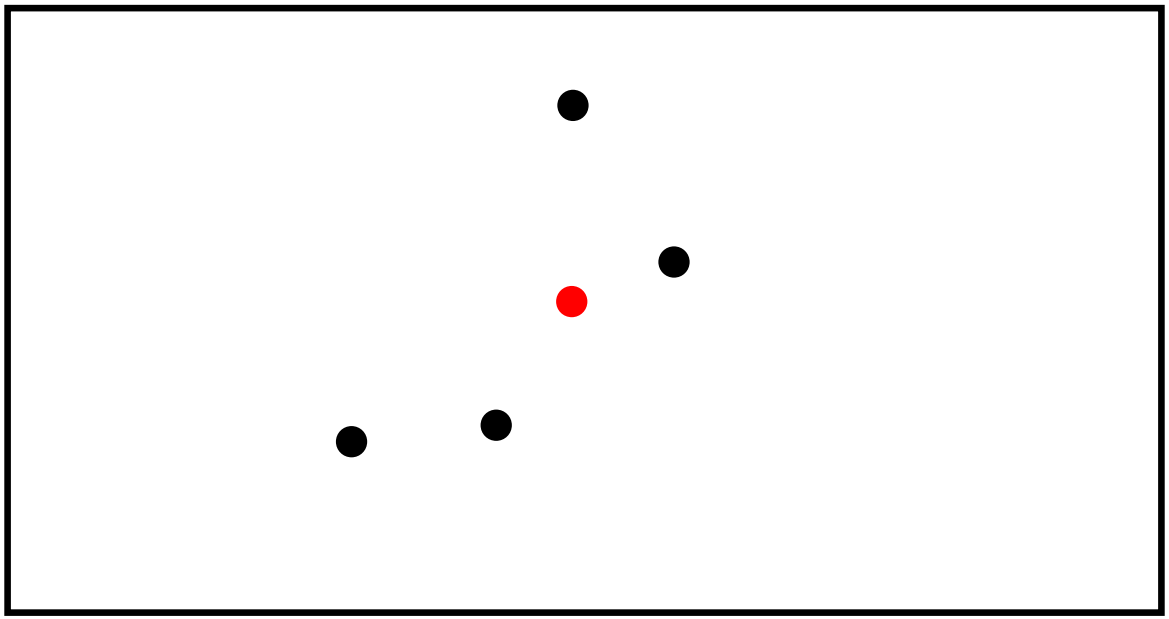
Exemple 2: partant de 2 ●, 3●



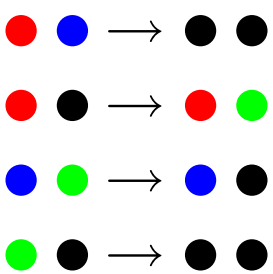
Program:



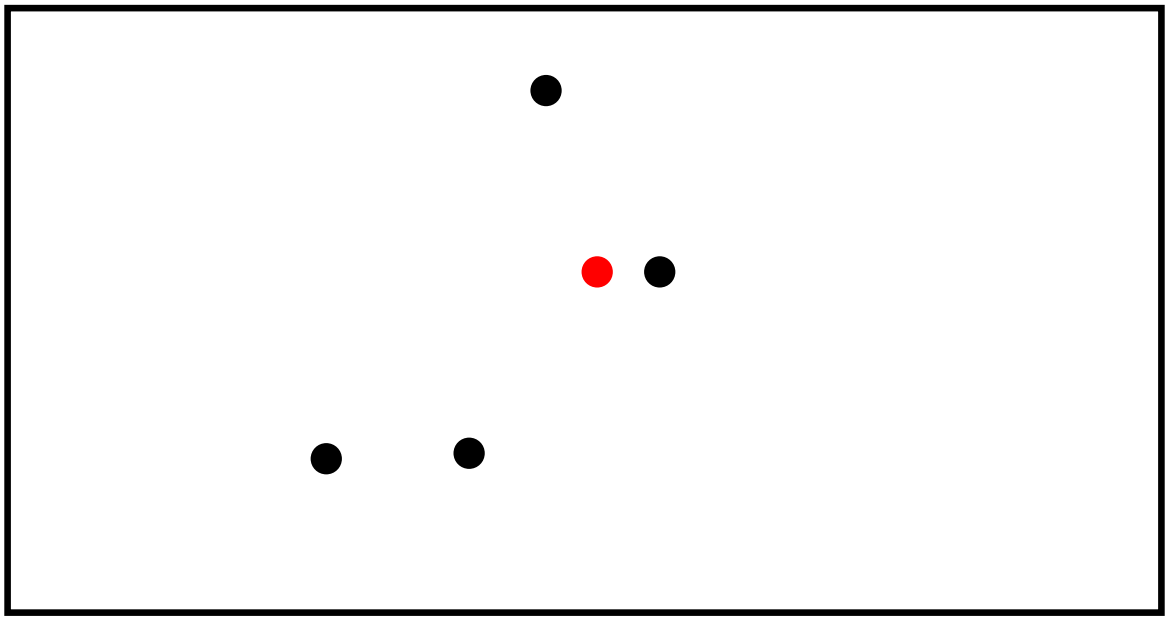
Exemple 2: partant de 2 ●, 3●



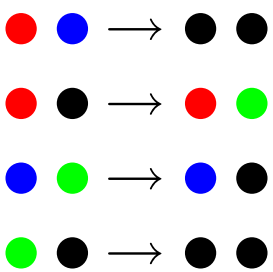
Program:



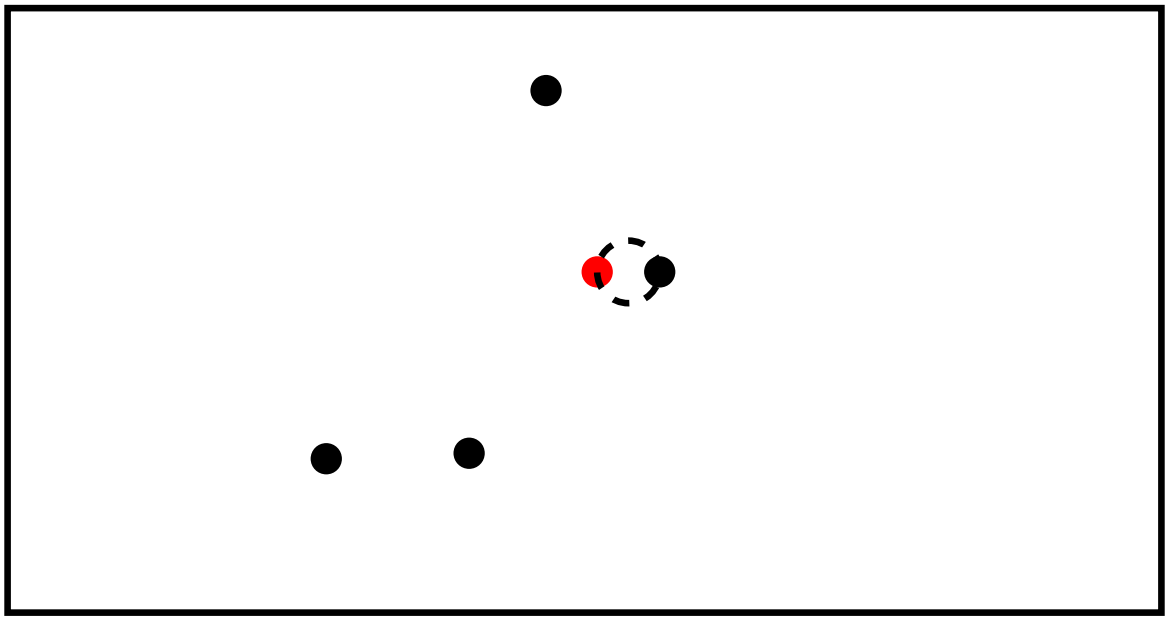
Exemple 2: partant de 2 ●, 3●



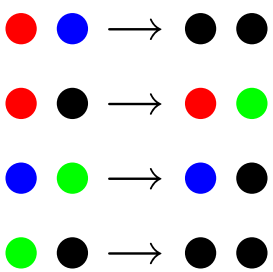
Program:



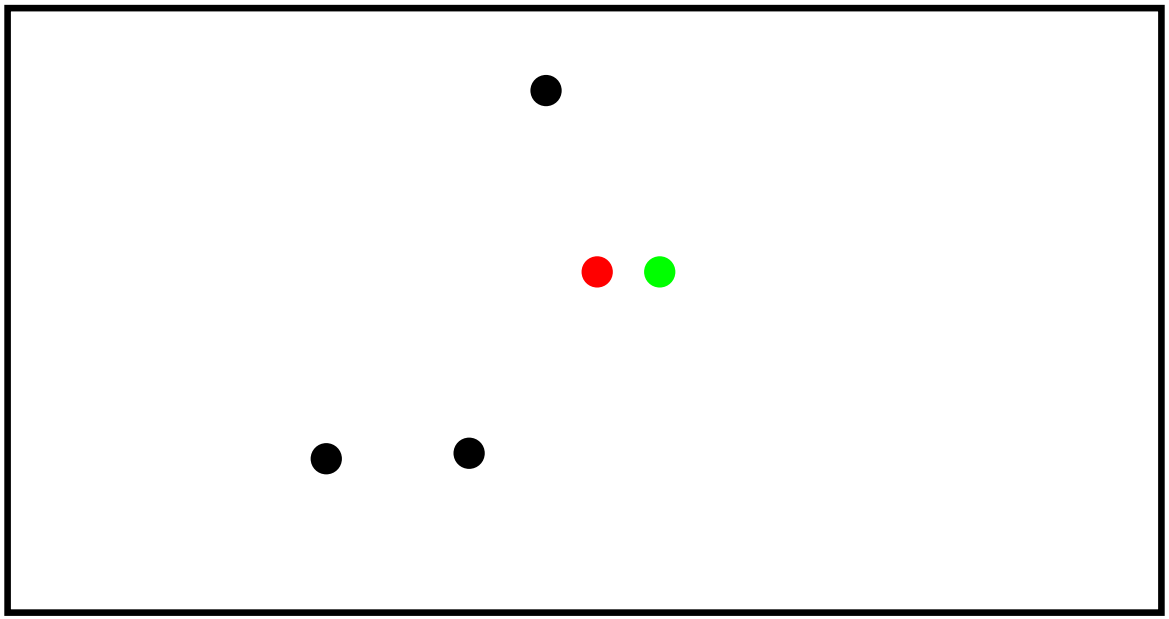
Exemple 2: partant de 2 ●, 3●



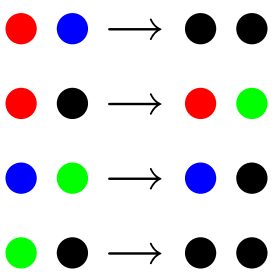
Program:



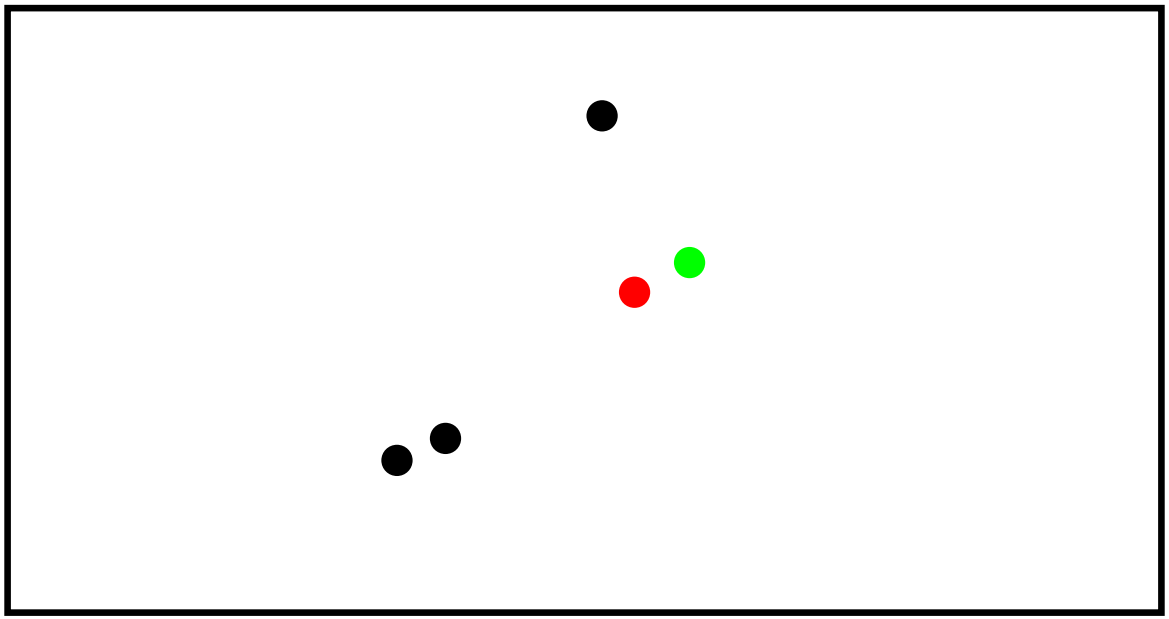
Exemple 2: partant de 2 ●, 3●



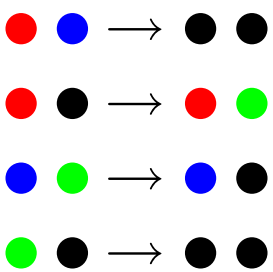
Program:



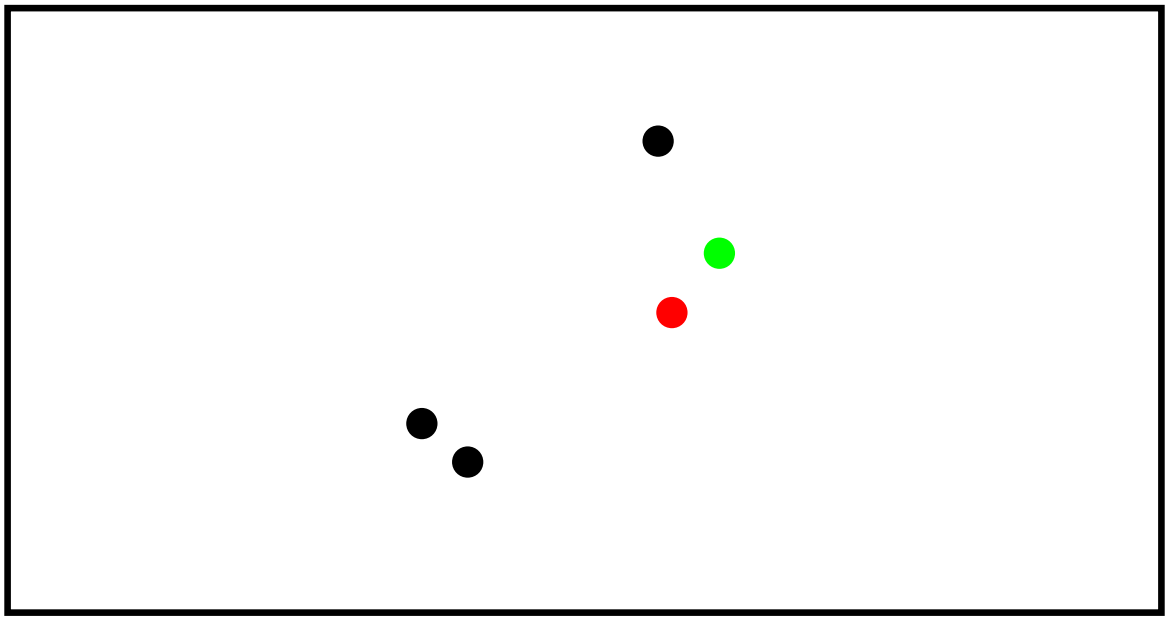
Exemple 2: partant de 2 ●, 3●



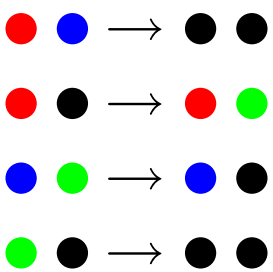
Program:



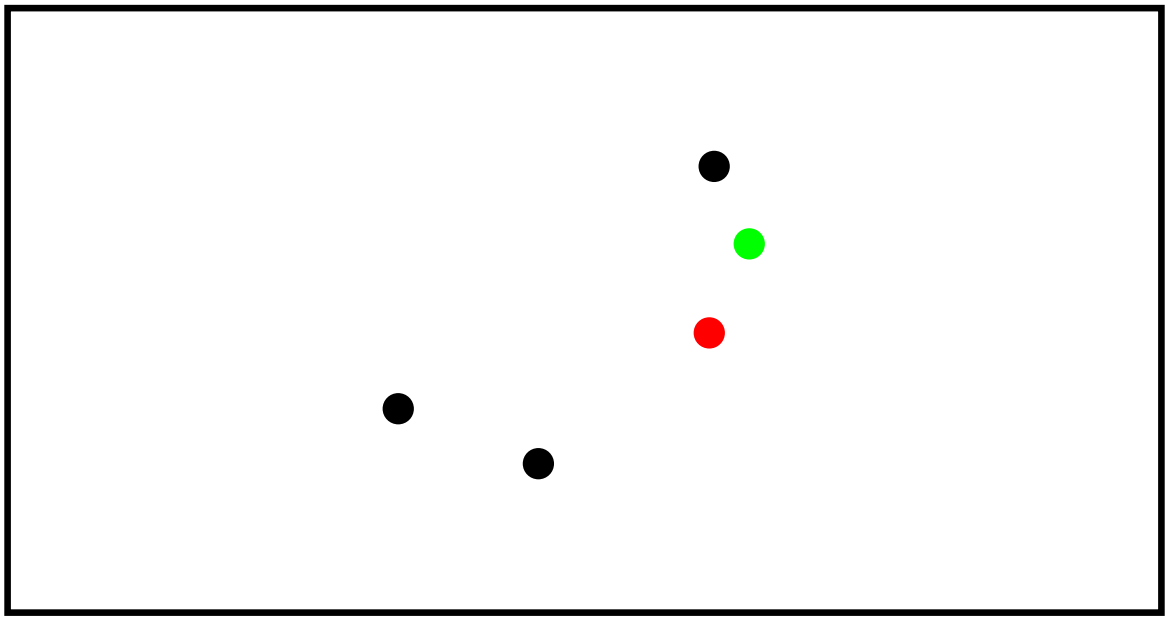
Exemple 2: partant de 2 ●, 3●



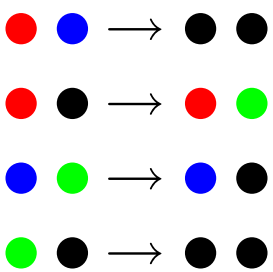
Program:



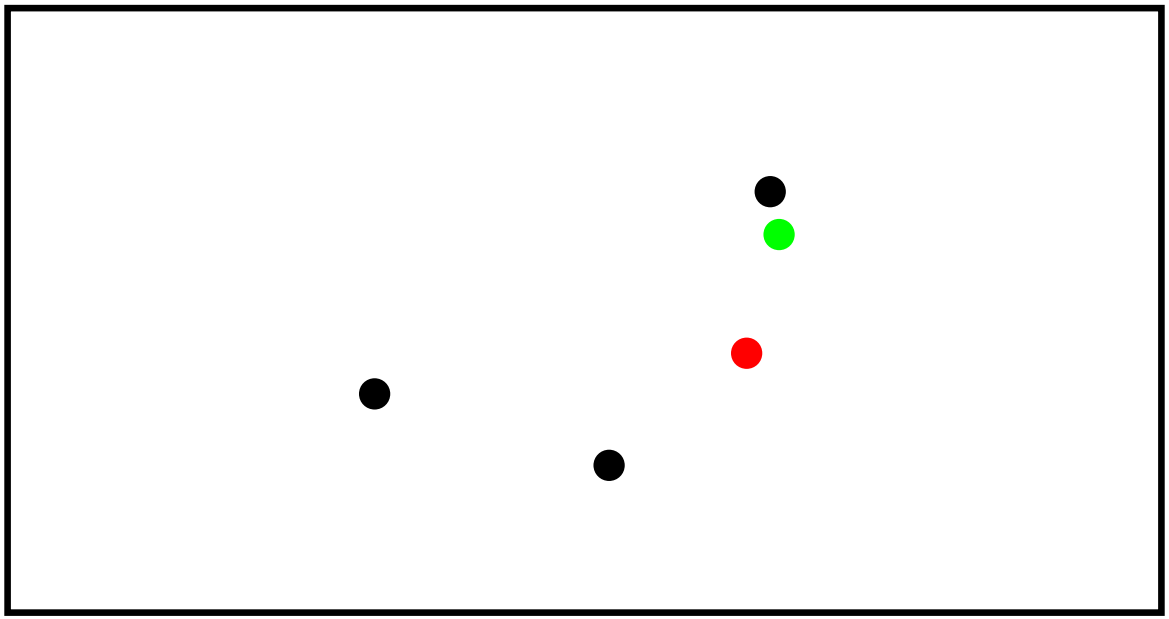
Exemple 2: partant de 2 ●, 3●



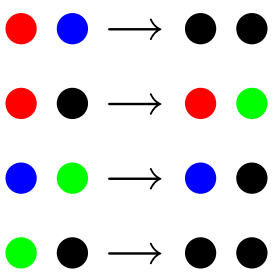
Program:



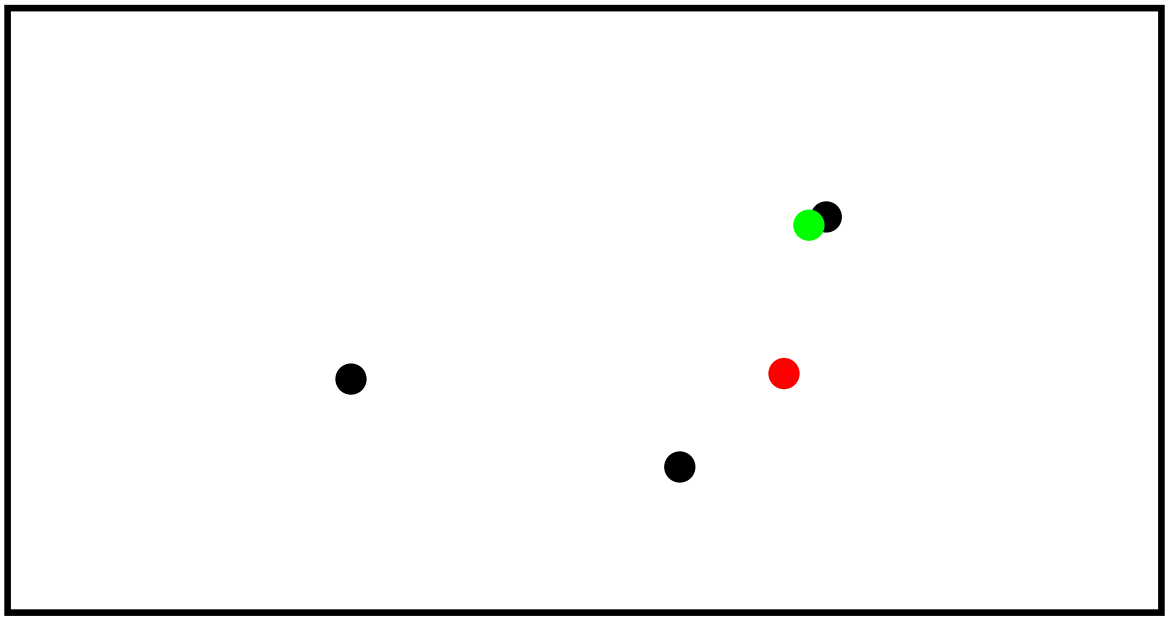
Exemple 2: partant de 2 ●, 3●



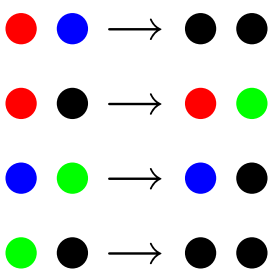
Program:



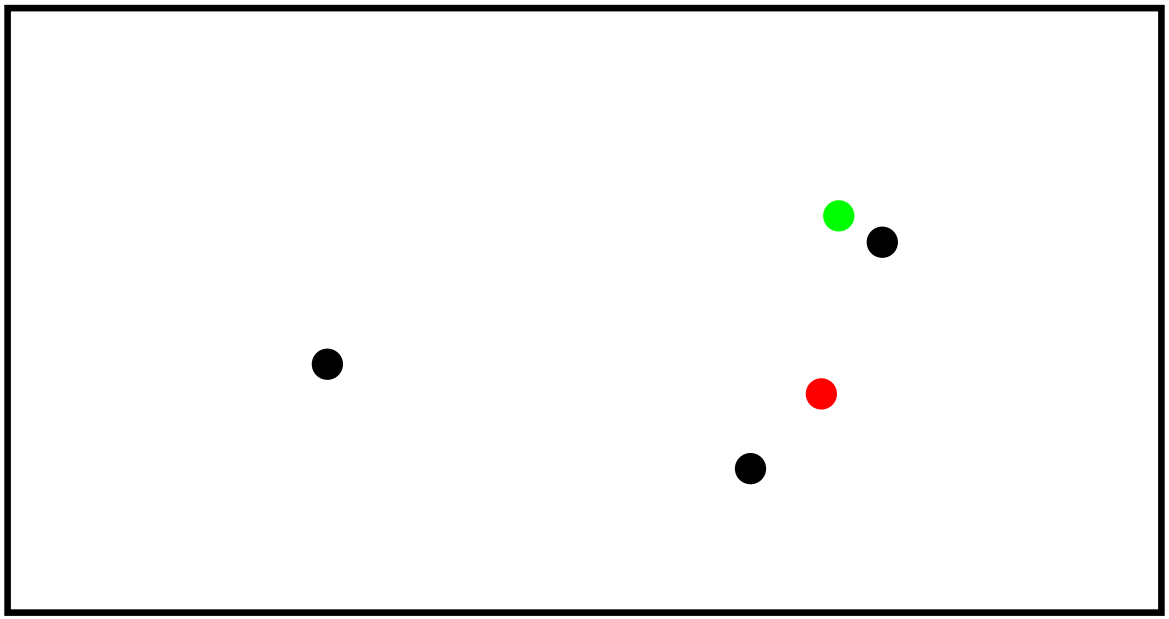
Exemple 2: partant de 2 ●, 3●



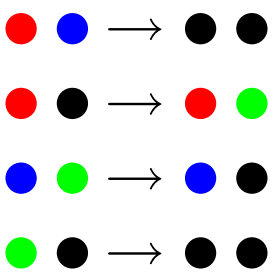
Program:



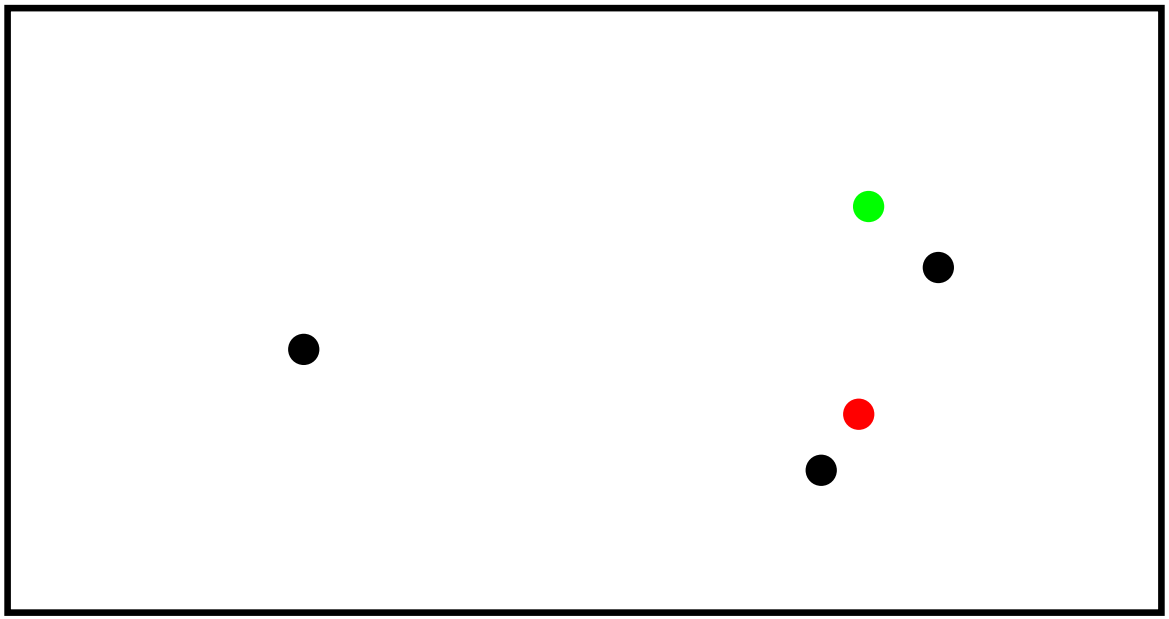
Exemple 2: partant de 2 ●, 3●



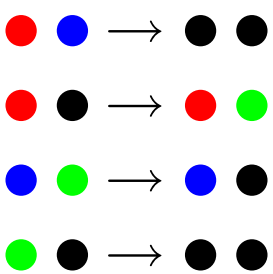
Program:



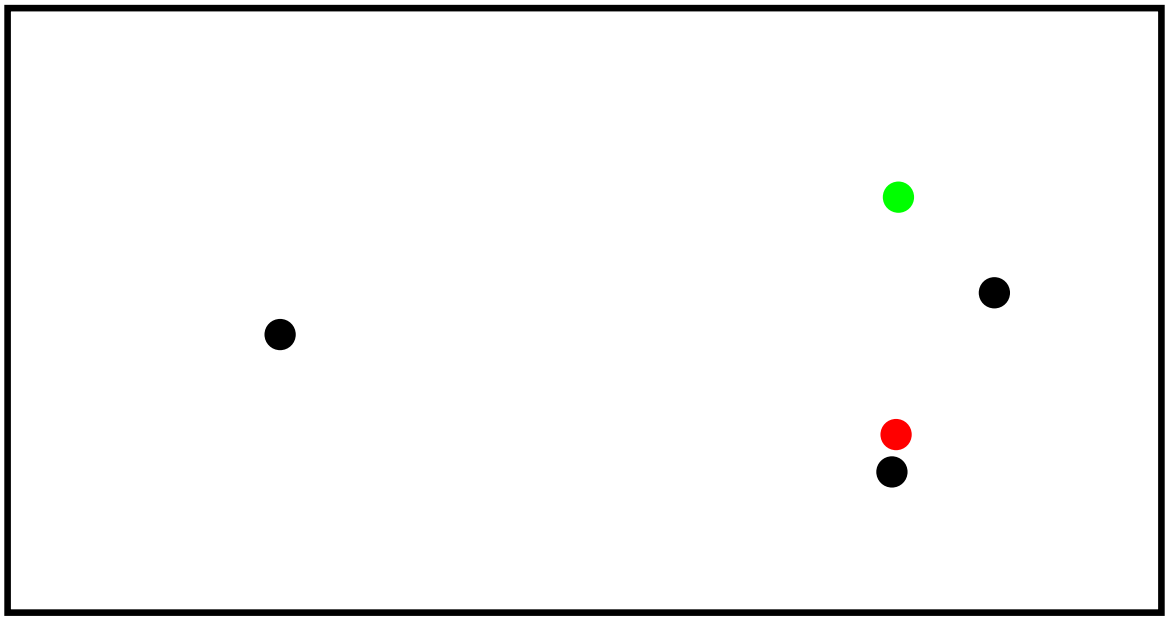
Exemple 2: partant de 2 ●, 3●



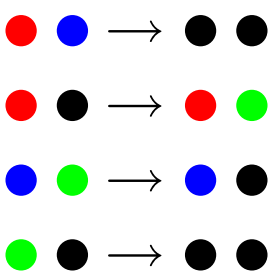
Program:



Exemple 2: partant de 2 ●, 3●

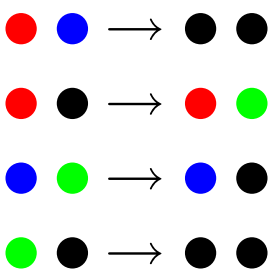


Program:



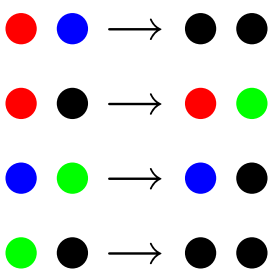
Exemple 2: partant de 2 ●, 3●

Program:



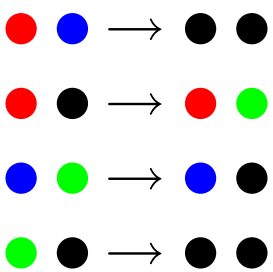
Exemple 2: partant de 2 ●, 3●

Program:



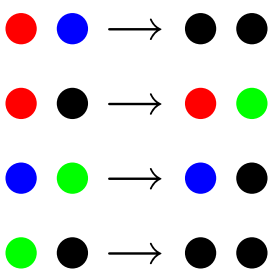
Exemple 2: partant de 2 ●, 3●

Program:



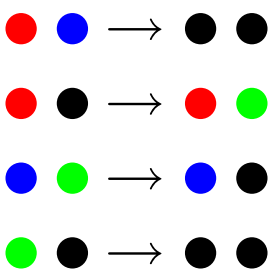
Exemple 2: partant de 2 ●, 3●

Program:



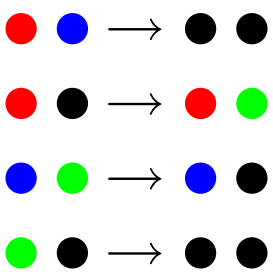
Exemple 2: partant de 2 ●, 3●

Program:

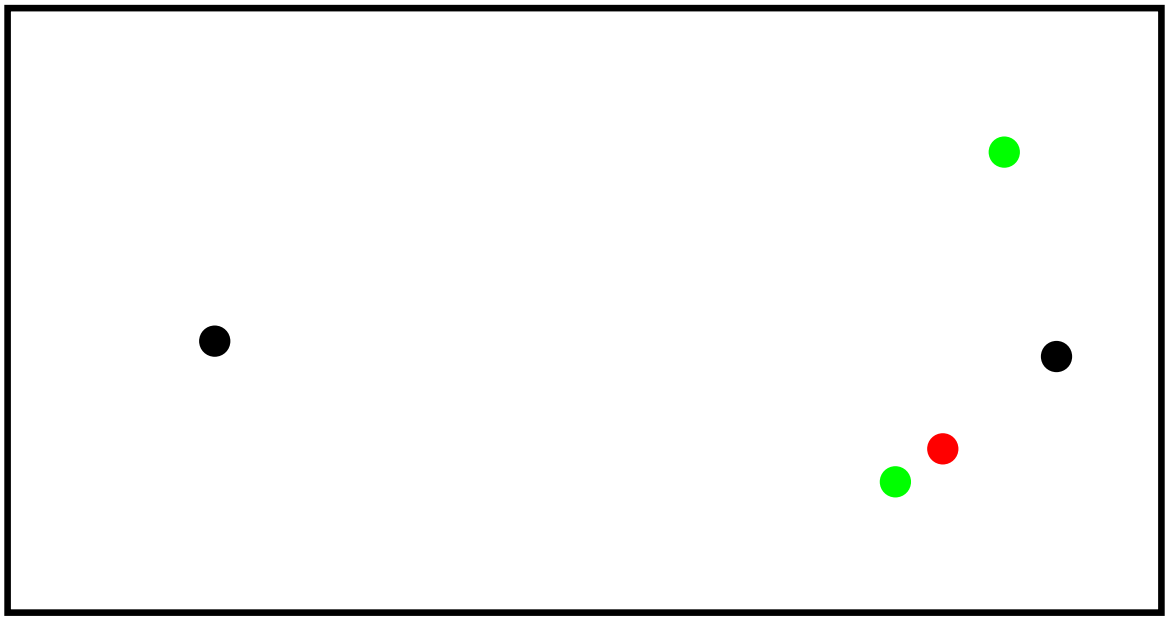


Exemple 2: partant de 2 ●, 3●

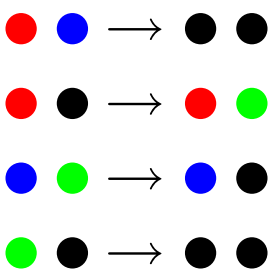
Program:



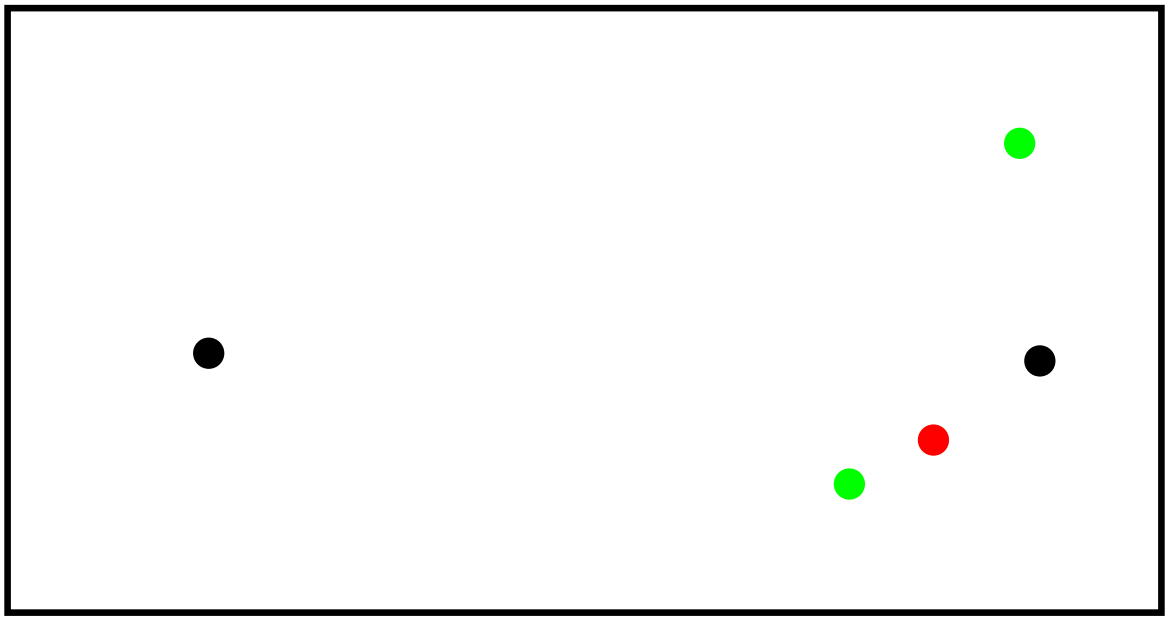
Exemple 2: partant de 2 ●, 3●



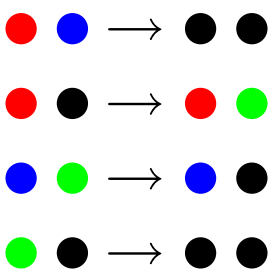
Program:



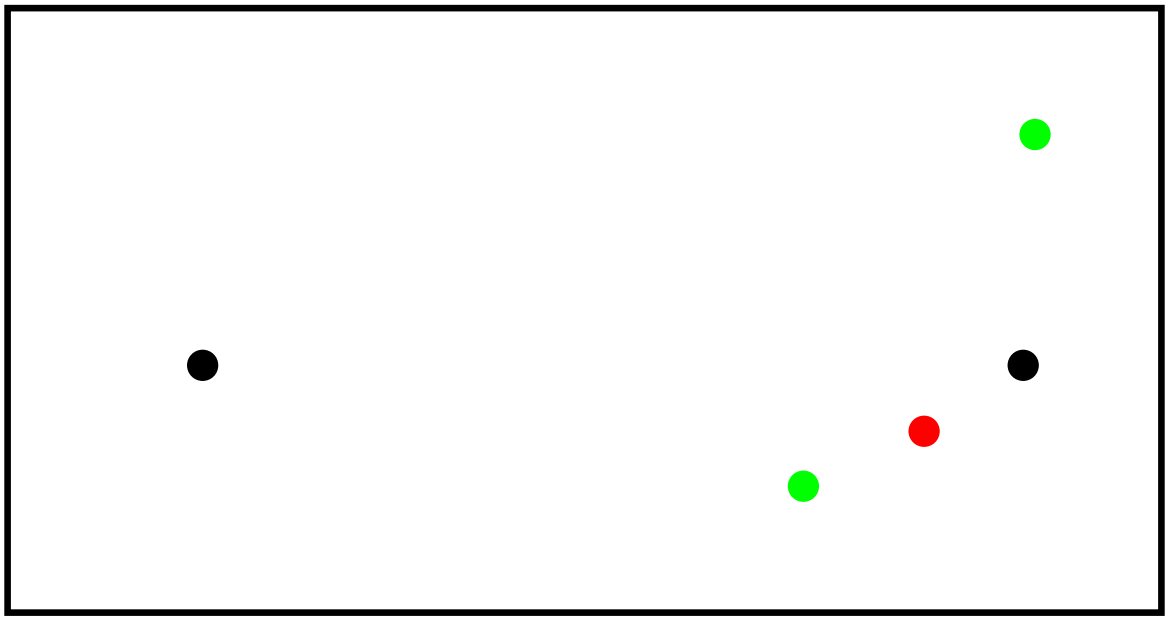
Exemple 2: partant de 2 ●, 3●



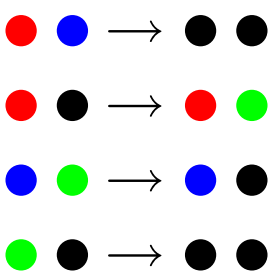
Program:



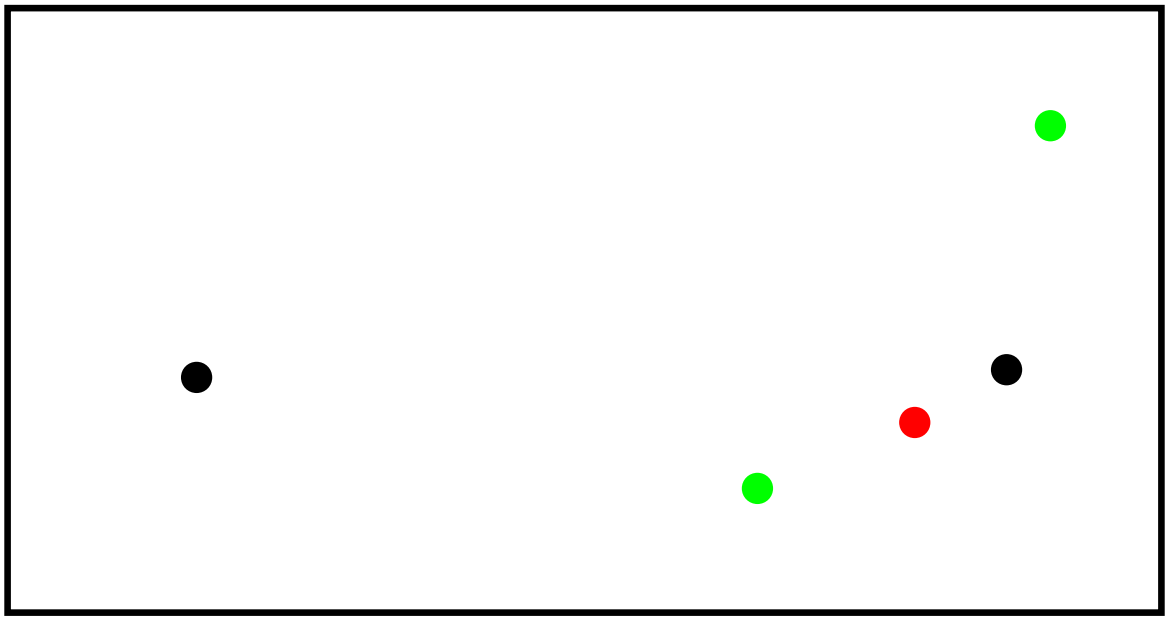
Exemple 2: partant de 2 ●, 3●



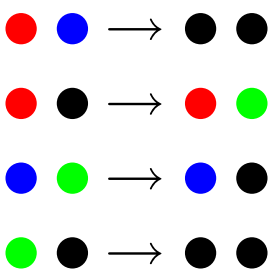
Program:



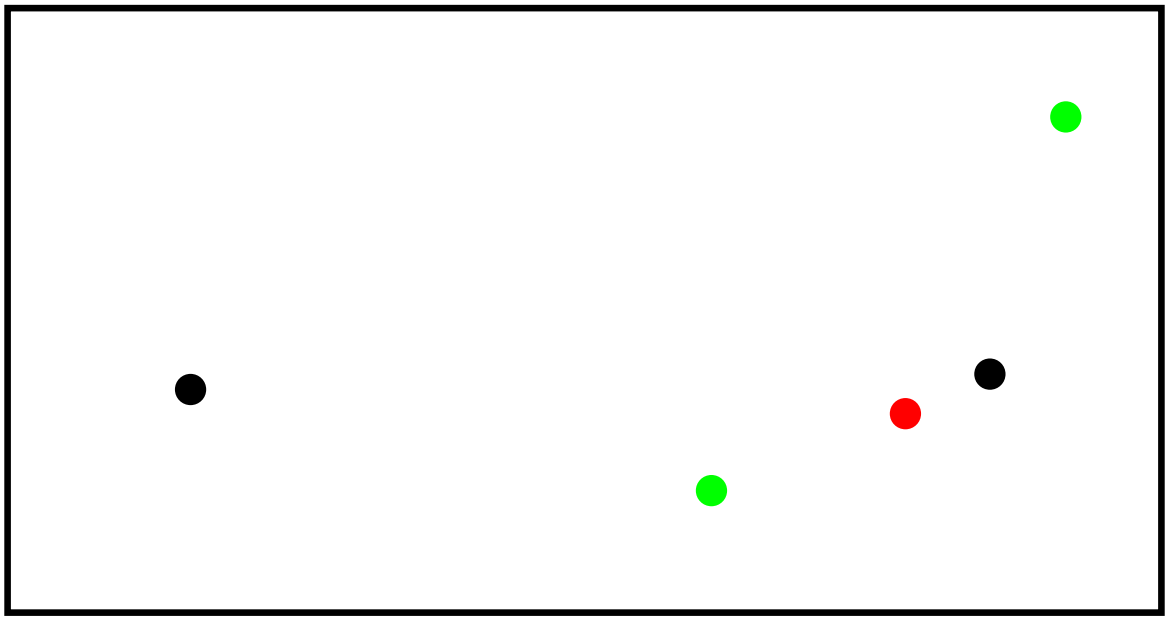
Exemple 2: partant de 2 ●, 3●



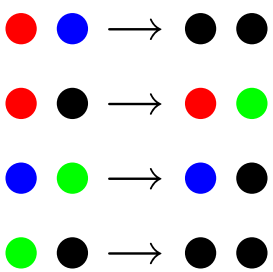
Program:



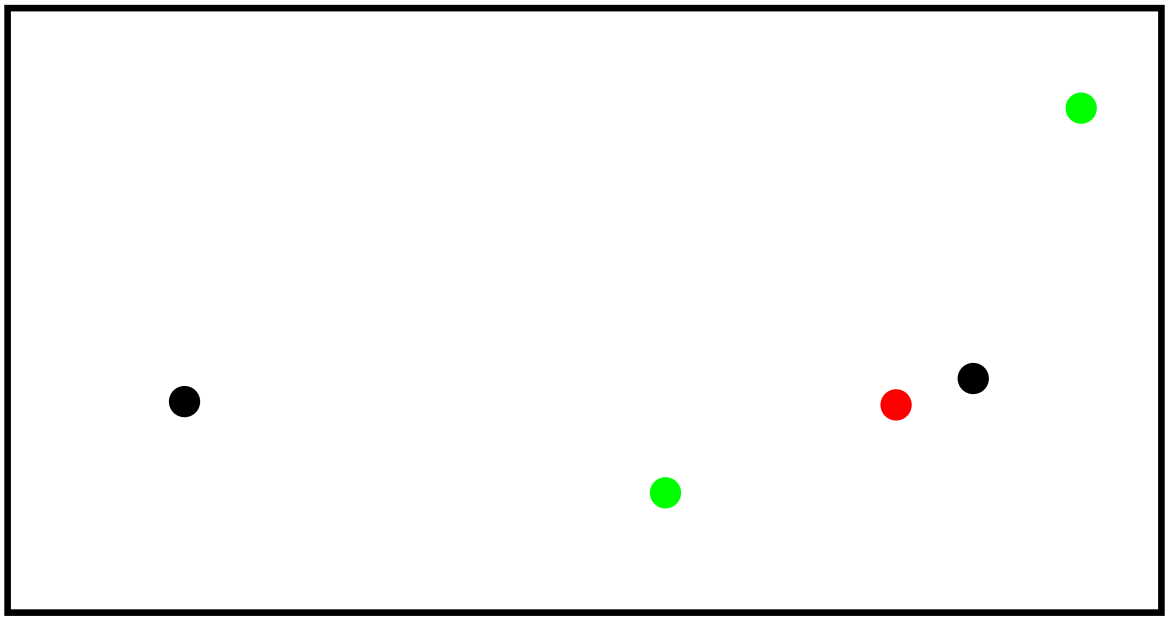
Exemple 2: partant de 2 ●, 3●



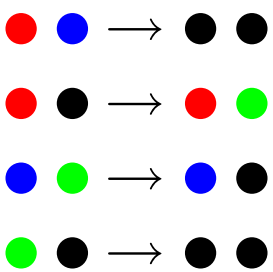
Program:



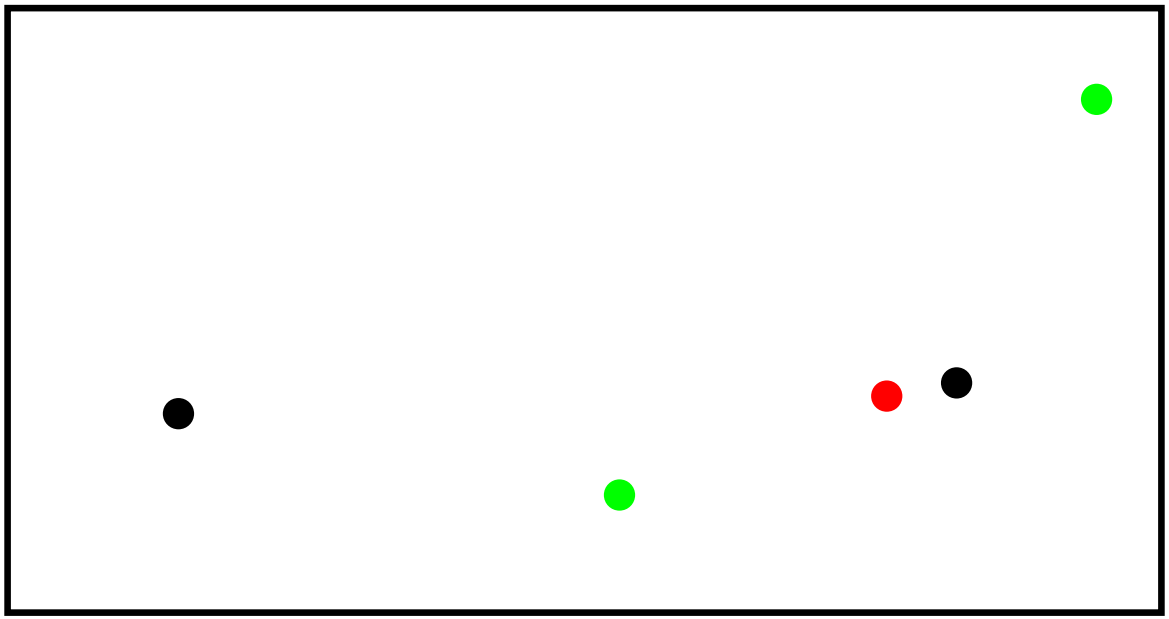
Exemple 2: partant de 2 ●, 3●



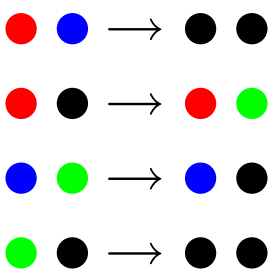
Program:



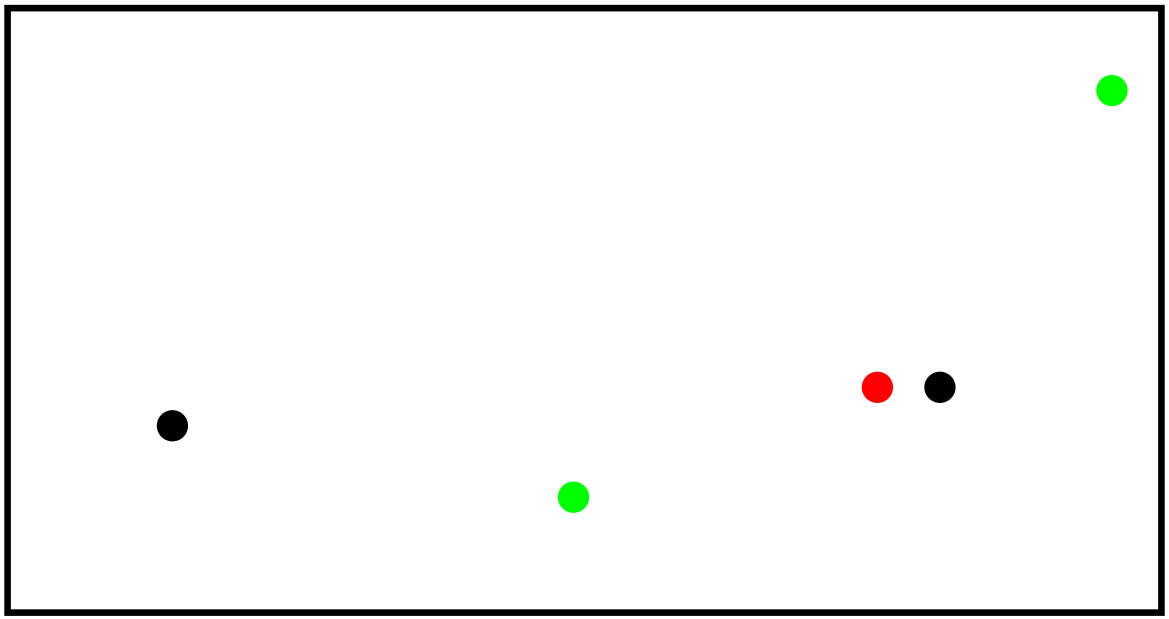
Exemple 2: partant de 2 ●, 3●



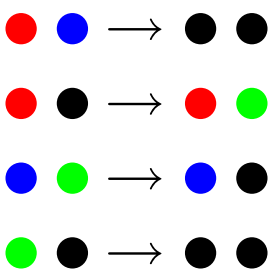
Program:



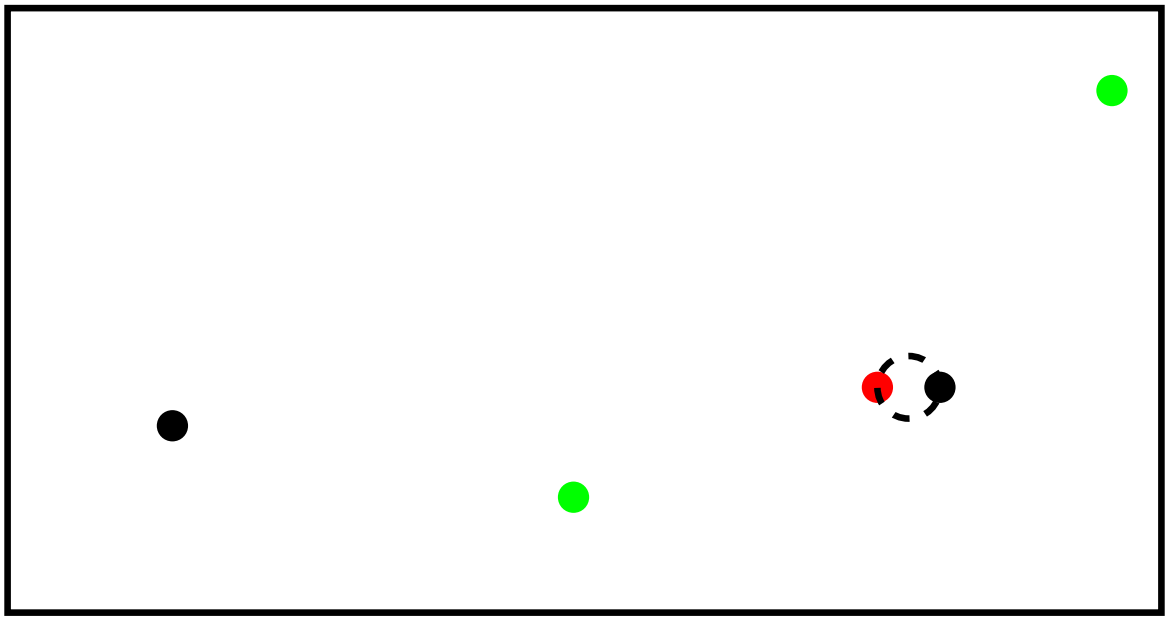
Exemple 2: partant de 2 ●, 3●



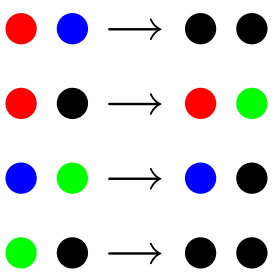
Program:



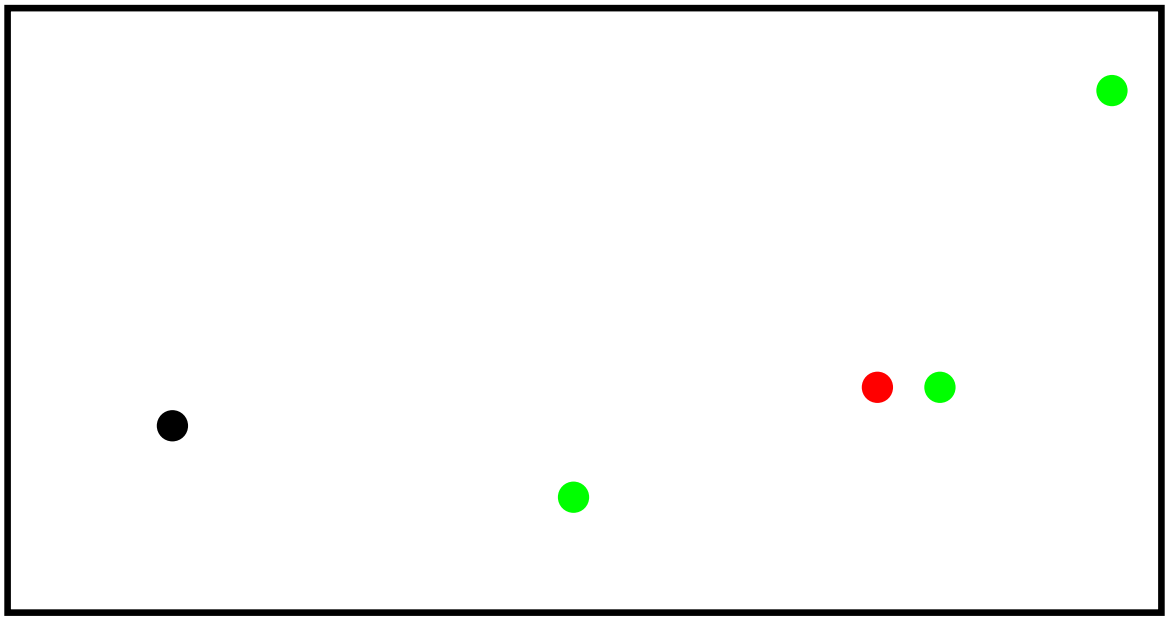
Exemple 2: partant de 2 ●, 3●



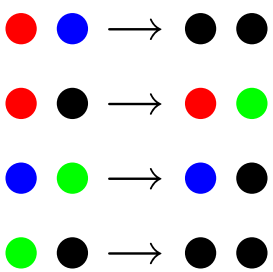
Program:



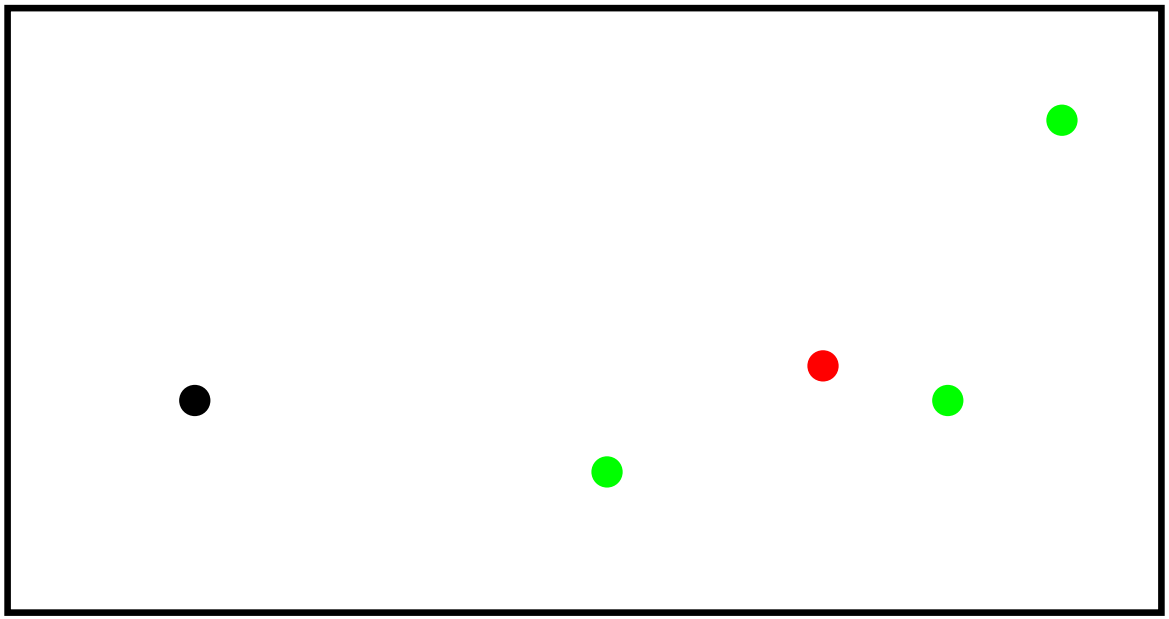
Exemple 2: partant de 2 ●, 3●



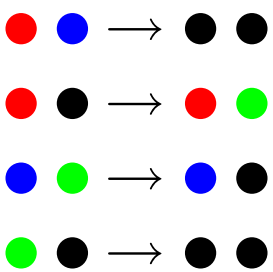
Program:



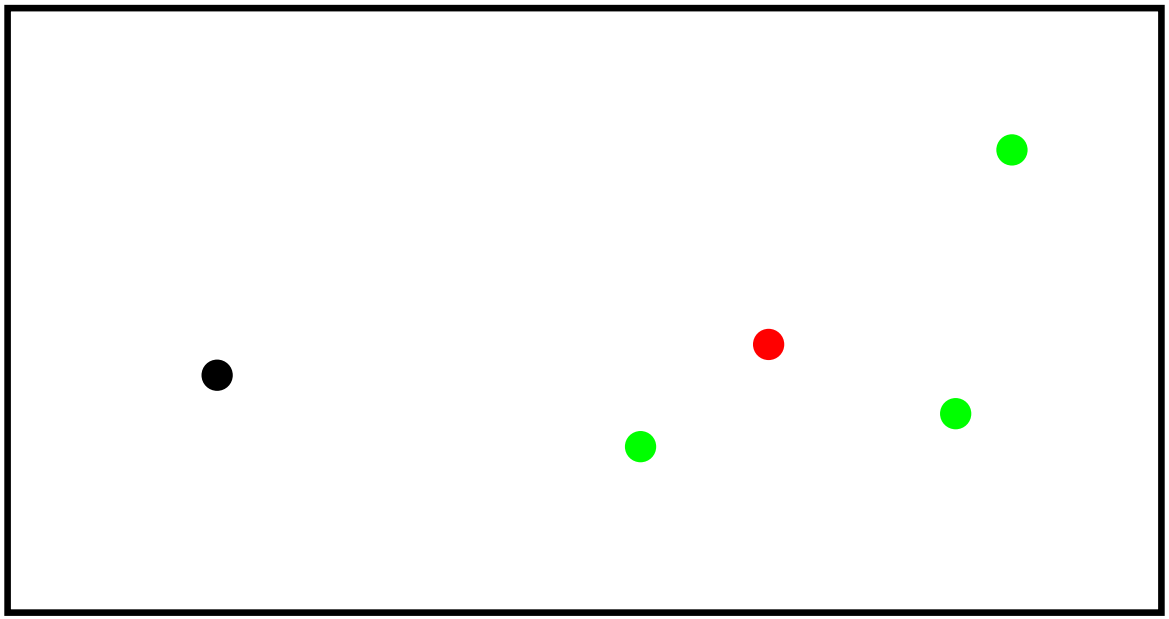
Exemple 2: partant de 2 ●, 3●



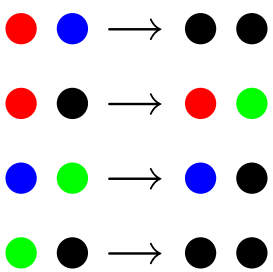
Program:



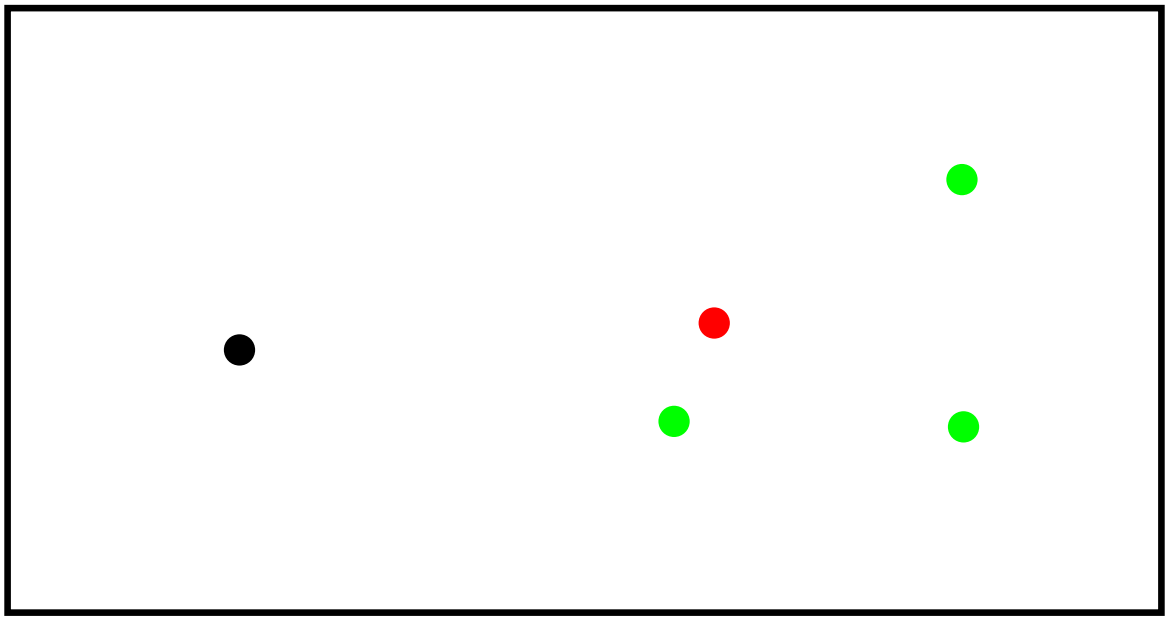
Exemple 2: partant de 2 ●, 3●



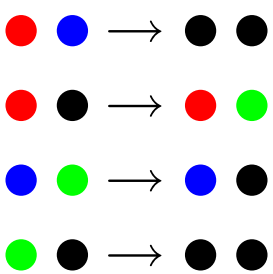
Program:



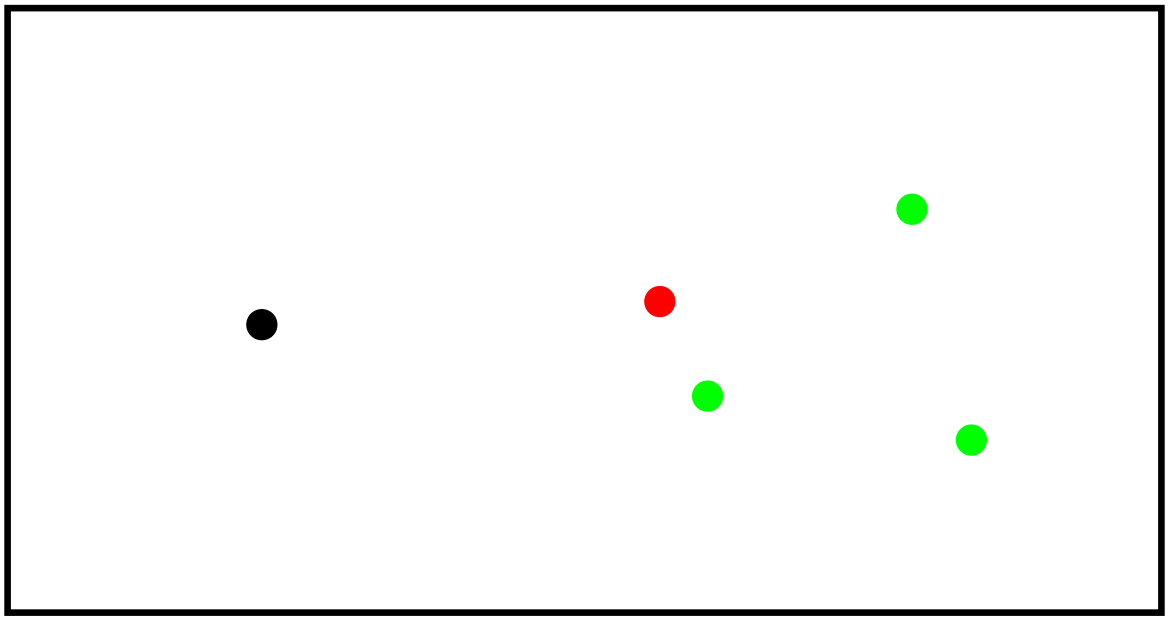
Exemple 2: partant de 2 ●, 3●



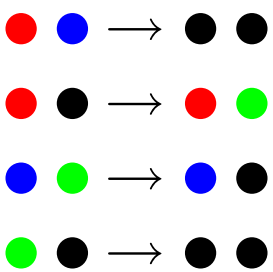
Program:



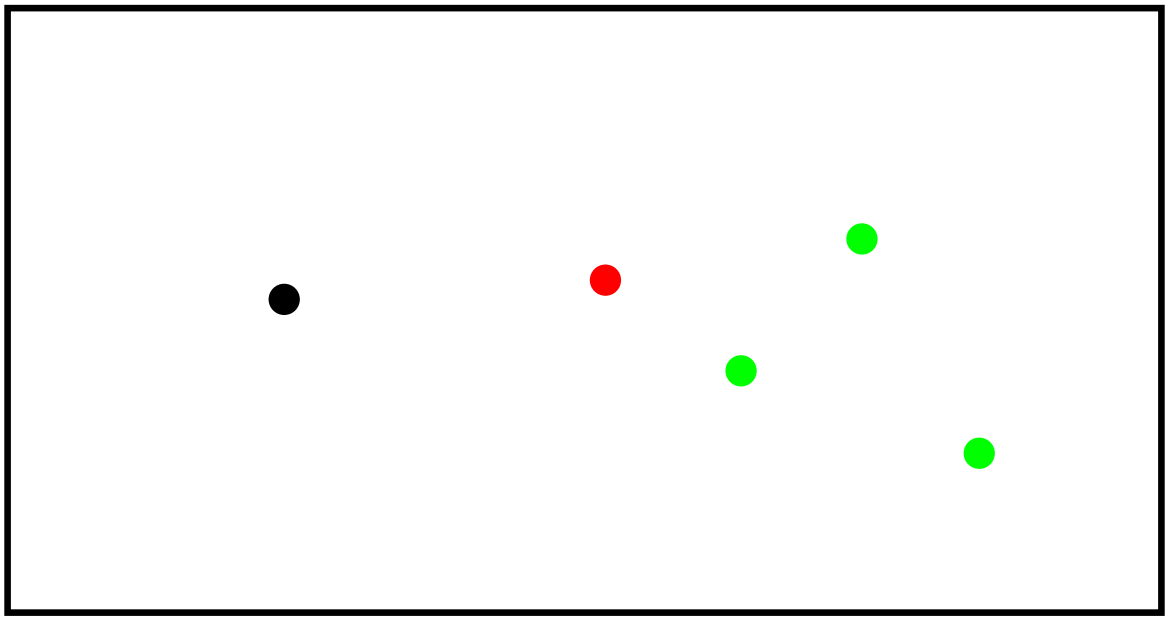
Exemple 2: partant de 2 ●, 3●



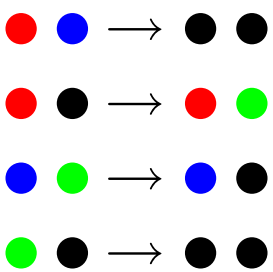
Program:



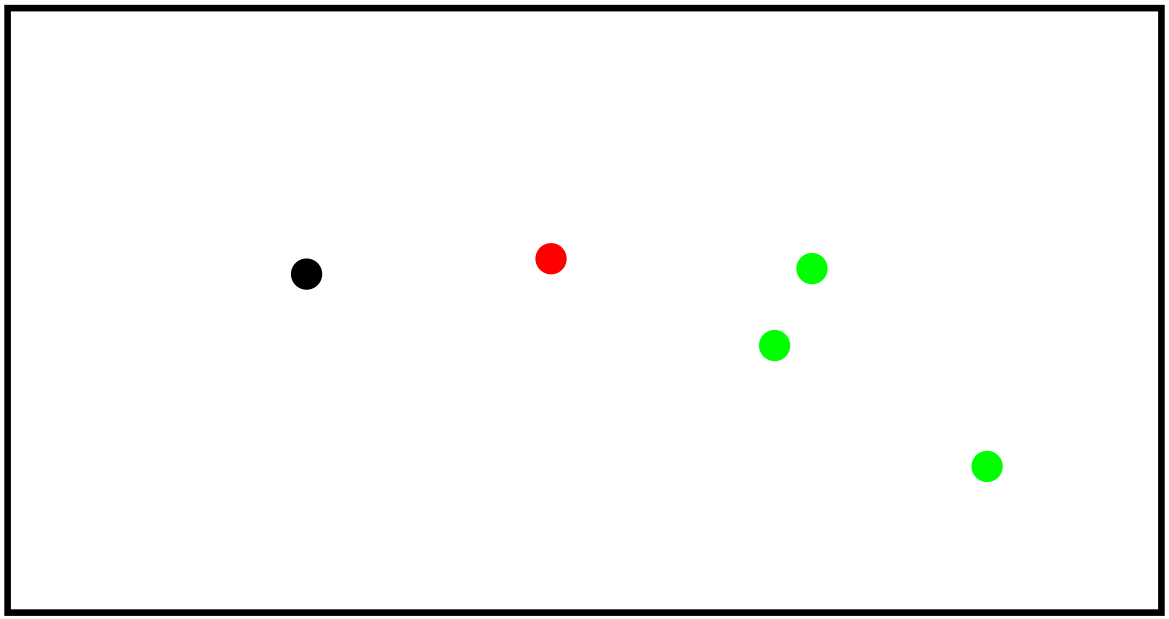
Exemple 2: partant de 2 ●, 3●



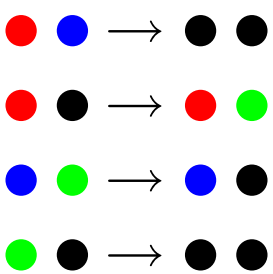
Program:



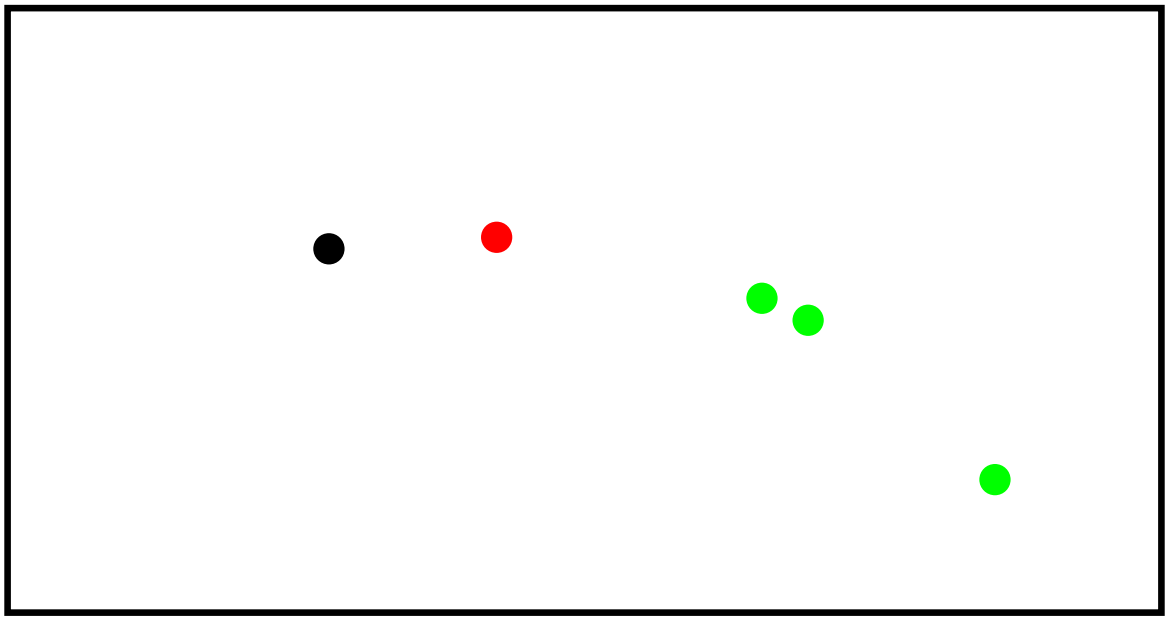
Exemple 2: partant de 2 ●, 3●



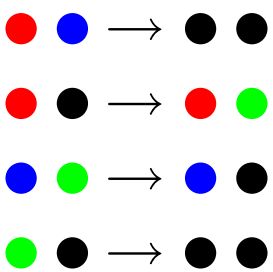
Program:



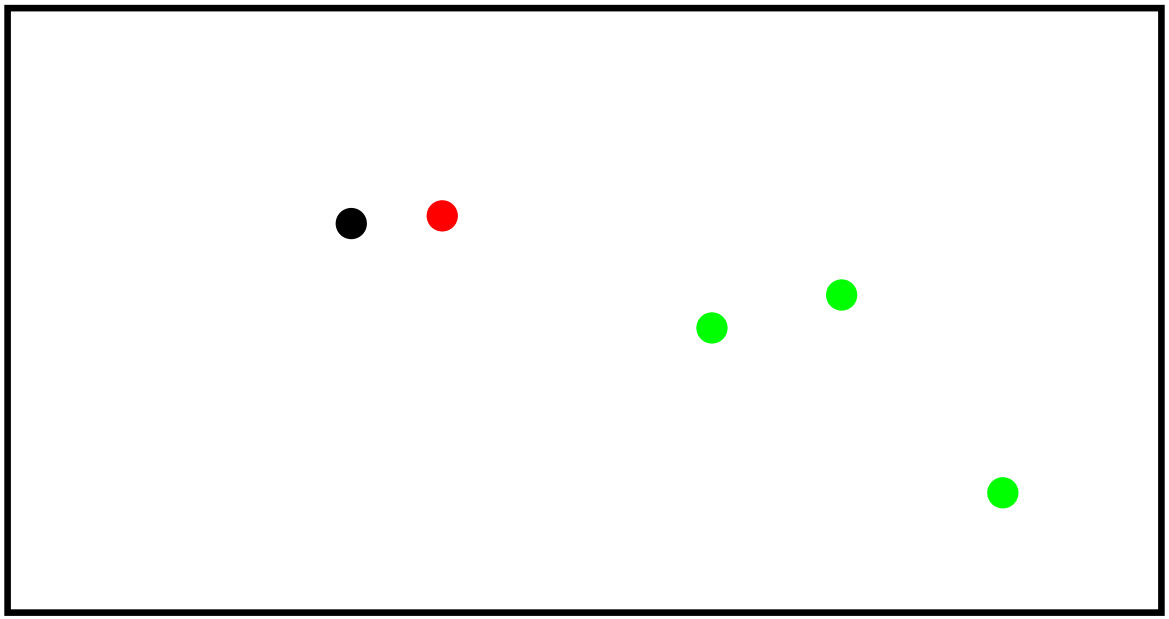
Exemple 2: partant de 2 ●, 3●



Program:



Exemple 2: partant de 2 ●, 3●



Program:

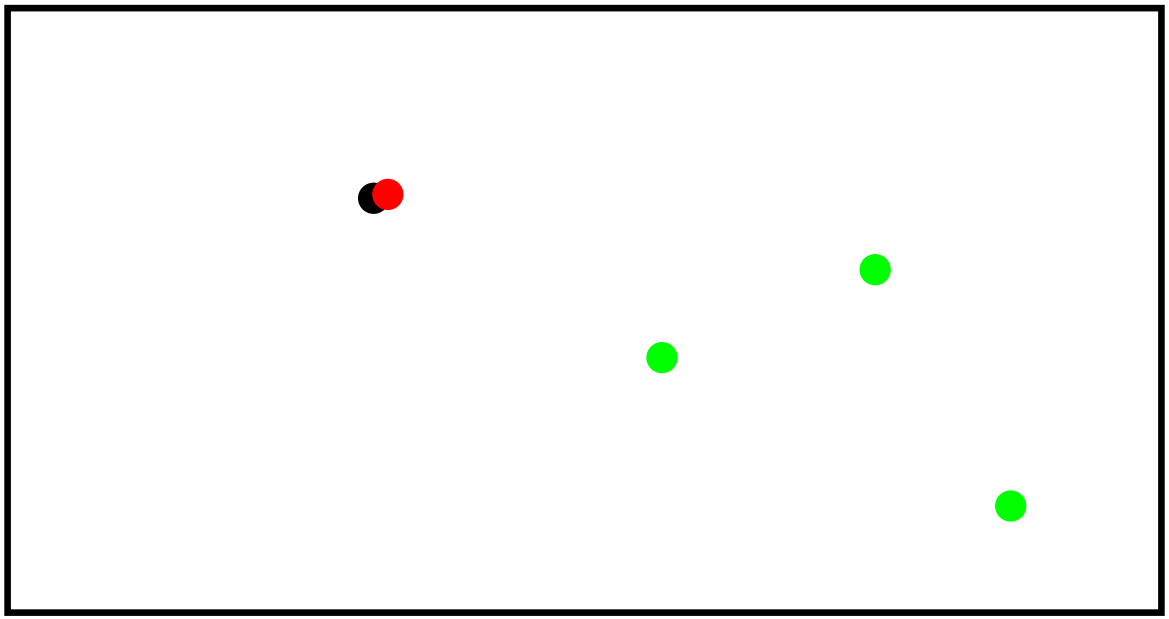
● ● → ● ●

● ● → ● ●

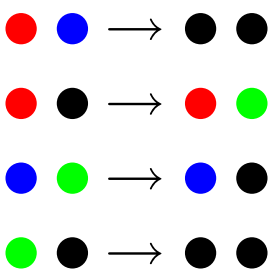
● ● → ● ●

● ● → ● ●

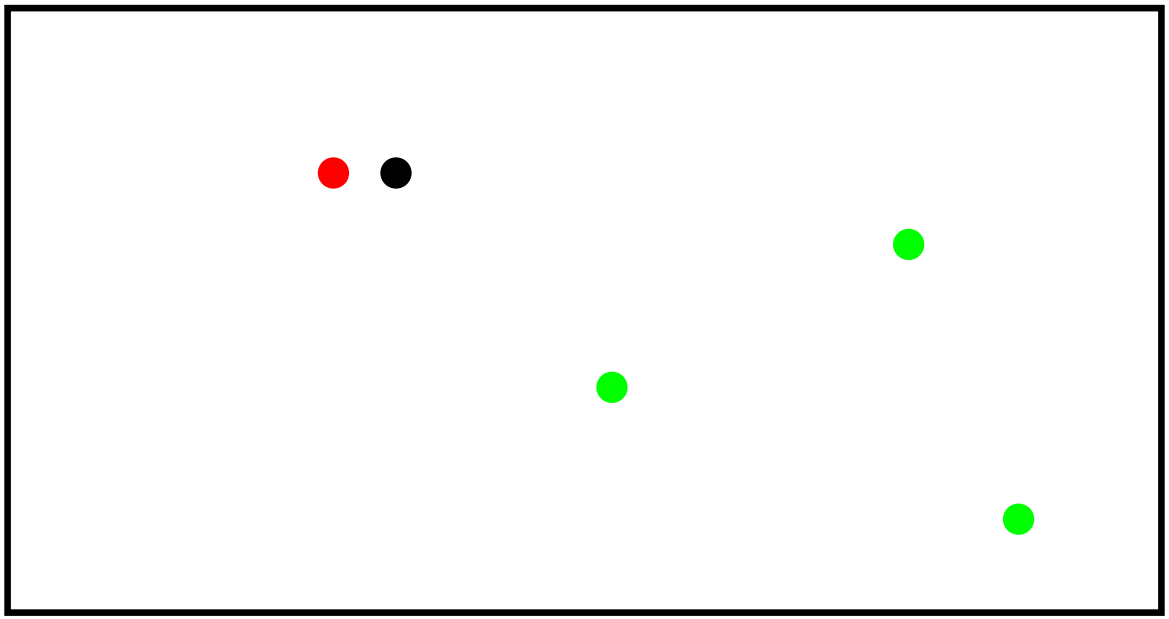
Exemple 2: partant de 2 ●, 3●



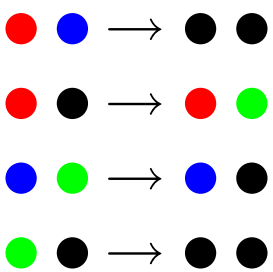
Program:



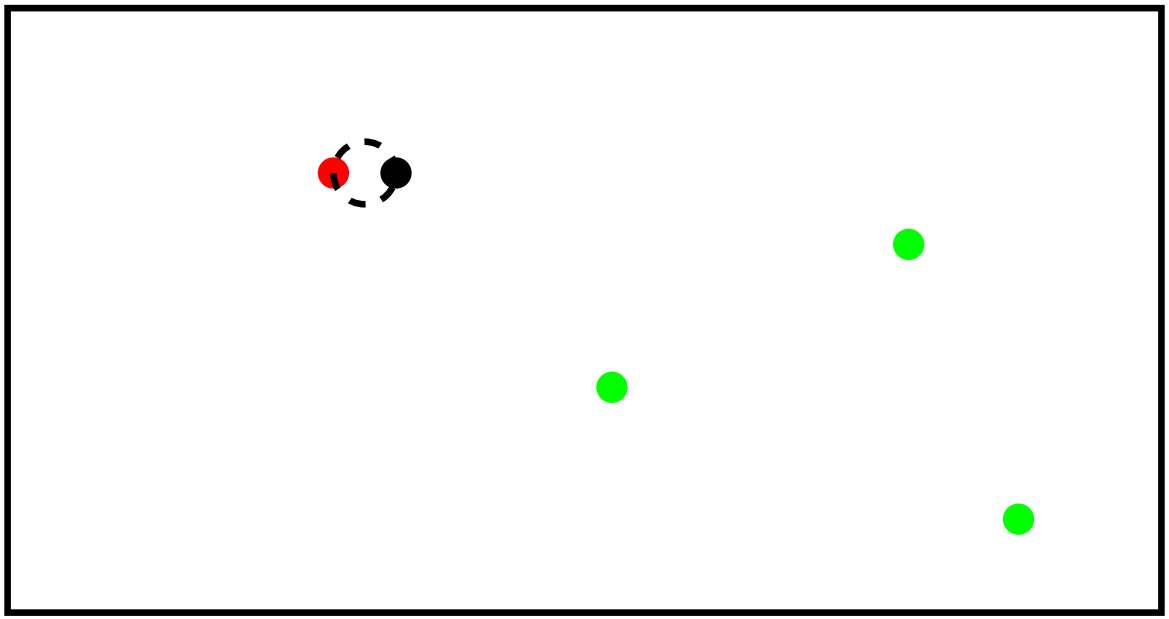
Exemple 2: partant de 2 ●, 3●



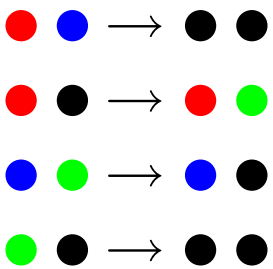
Program:



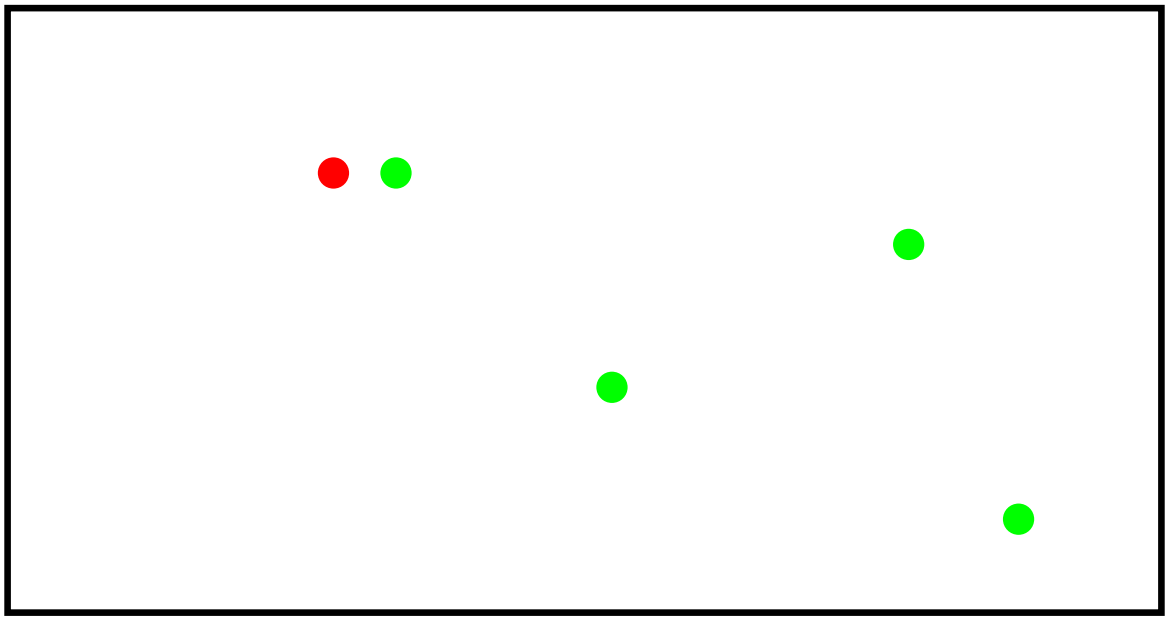
Exemple 2: partant de 2 ●, 3●



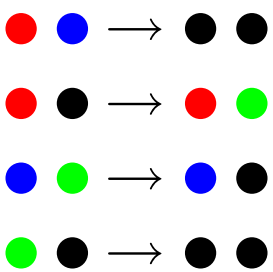
Program:



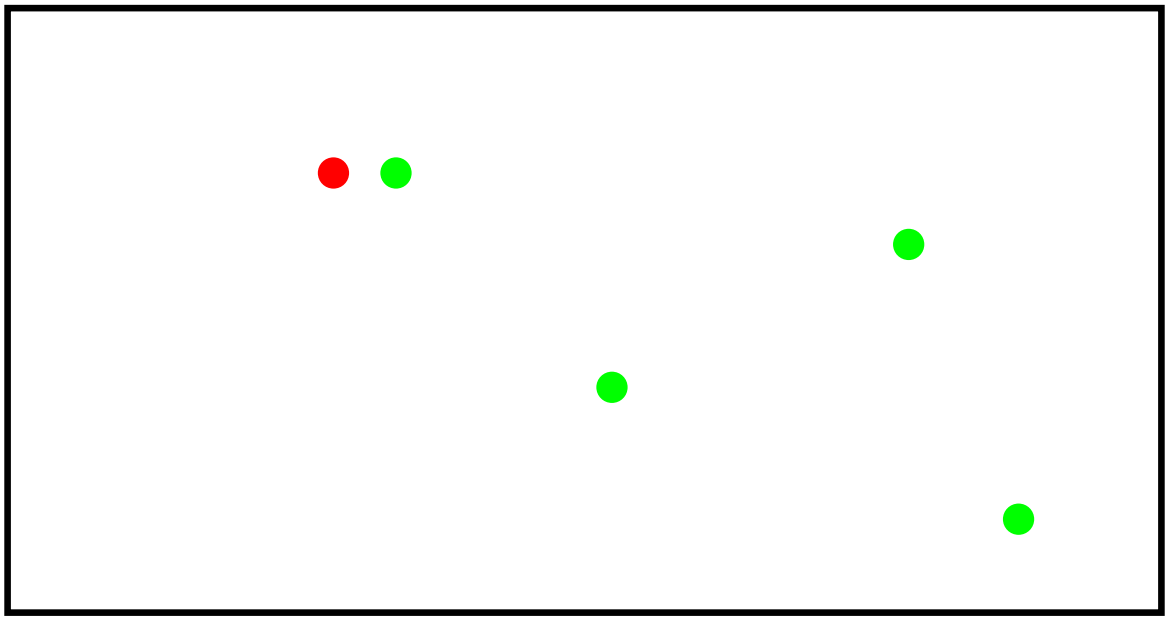
Exemple 2: partant de 2 ●, 3●



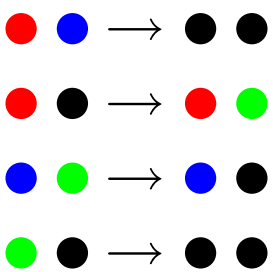
Program:



Example 2: Résultat final



Program:



What is computed?

Let's interpret

- ●and ●by yes.
- ●and ●by no.
- Whatever the initial state is, ultimately, all agents will agree.
- They agree on *yes* iff the initial population of ●is strictly greater than the initial population of ●.

Alternative statement

- A configuration can be seen as an element of $\mathbb{N}^4$.
  - (n_b, n_r, n_g, n_{bl}) if there is n_b ●, n_r ●, n_g ●and n_{bl} ●
  - $n_b + n_r + n_g + n_{bl}$ is preserved.
- An initial configuration is of type $(n_b, n_r, 0, 0)$.
- This protocol computes predicates $n_r > n_b$, i.e.

MAJORITY .

General Case: Algorithm

An algorithm consists of

- a finite set of states $Q = \{q_1, q_2, \cdots, q_k\}$.
- transition rules, mapping pairs of states to pairs of states

Executions are given by:

- instantaneous configuration: a multiset of states = an element of $\mathbb{N}^k$.
- transitions between configuration: two agents are picked, and updated according to the rules of the algorithm.
- output interpretation: mapping states to $\{0, 1\}$.
- a computation is over when all agents agree on 0 or 1.

Remark: Algorithm are assumed independent of size of population!

Simplest example: Computing OR of input bits

States: ●,●

One transition rule: ● ● → ● ●

Output of an agent is its state.

If all inputs are ●, all agents will remain in state ●
If some agent is ●, eventually all will have state ●

Simplest example: Computing OR of input bits

States: ●,●

One transition rule: ● ● → ● ●

Output of an agent is its state.

If all inputs are ●, all agents will remain in state ●
If some agent is ●, eventually all will have state ●

A remark: Fairness

- One need to guarantee that all possible interactions happen eventually
  - an execution is *fair* if for all configurations C that appear infinitely often in the execution, if $C \longrightarrow C'$ for some configuration C', then C' appears infinitely often in the execution.
  - can be seen as capturing the idea of probabilistic adversary: there is some (unknown) underlying probability distribution on interactions such that events are independent.

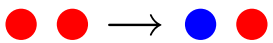
- True notion of computation: For any input I , for any fair sequence of executions starting from I agents ultimately agree on 0 or 1.

Leader Election

Initially, all agents in same state

Eventually, exactly one agent is in a special leader state

Program:



Threshold Predicate

- Suppose each agent starts with input ● or ●
- Determine whether at least five agents have input ●.

5%

- Each agent is initially ● or ●
- Determine whether at least 5% of the inputs are ●

- Similar to majority, except each ● can cancel 19 ●'s.

5%

- Each agent is initially ● or ●
- Determine whether at least 5% of the inputs are ●
- Similar to majority, except each ● can cancel 19 ●'s.

40%

- Each agent is initially ● or ●
- Determine whether at least 5% of the inputs are ●

- A bit trickier (exercice).

40%

- Each agent is initially ● or ●
- Determine whether at least 5% of the inputs are ●
- A bit trickier (exercice).

How to Compute $\sum_{i=1}^{k} c_i x_i \geq a$

Input convention: each agent with ith input symbol starts in state c_i.
Each agent has a *leader bit* governed by ● ● → ● ●.
Let $m = \max(|a| + 1, |c_1|, ..., |c_k|)$.
Each agent also stores a value from $-m, -m+1, ..., m-1, m$.
If a leader meets a non-leader, their values change as follows:

$$\begin{array}{ll} x, y \rightarrow x + y, 0 & \text{if } 0 \leq x + y \leq m \\ x, y \rightarrow m, x + y - m & \text{if } x + y > m \\ x, y \rightarrow -m, x + y + m & \text{if } x + y < -m \end{array}$$

(In each case first agent on right hand side is the leader.)

Each agent also remembers output of last leader it met.

Correctness

Sum of agents' values is invariant.

Sum is eventually gathered into the unique leader
(up to maximum absolute value of m):

If $sum > m$, leader has value m $\Rightarrow$ Output Yes.
If $sum < -m$, leader has value -m $\Rightarrow$ Output No.
If $-m \leq sum \leq m$, leader's value is the actual sum $\Rightarrow$ Output depends on sum.

In each case, leader knows output and tells everyone else.

Computable Predicates

Theorem (Angluin et al. 2006)

A predicate is computable iff it is on the following list.

- $\sum_{i=1}^{k} c_i x_i \geq a$, *where* a, c_i*'s are integer constants*
- $\sum_{i=1}^{k} c_i x_i \equiv a \pmod{b}$ *where* a, b *and* c_i*'s are constants*
- *Boolean combinations of the above predicates*

Alternate Characterization: Presburger Arithmetic

A predicate is computable iff it can be expressed in first-order logic using the symbols $+, 0, 1, \vee, \wedge, \neg, \forall, \exists, =, <, (,)$ and variables.

(This system is known as Presburger Arithmetic [1929].)

Note: no multiplication.

Examples:

majority: $x_0 < x_1$

divisible by 3: $\exists y : y + y + y = x_1$

at least 40%:$x0 + x0 < x1 + x1 + x1$

Alternate Characterization: Semilinear Sets

A predicate is computable iff it the set of inputs with output yes is semilinear.

A set of vectors $\vec{x} = (x_1, x_2, ..., x_k) \in \mathbb{N}^k$ is linear if it is of the form

$$\{\vec{v_0} + c_1\vec{v_1} + c_2\vec{v_2} + \ldots + c_m\vec{v_m} : c_1, \ldots, c_m \in \mathbb{N}\}$$

A set of vectors is semilinear if it is a finite union of linear sets.

Variants of the model

The basic classical model is now already understood pretty well.

Variants considered in literature:

- Limited interaction graph
- One-way communication
- Failures

Dana Angluin, James Aspnes, Melody Chan, Carole Delporte-Gallet, Zoë Diamadi, David Eisenstat, Michael J. Fischer, Hugues Fauconnier, Rachid Guerraoui, Hong Jiang, René Peralta, Eric Ruppert

Plan

One-way communication

- Classical model assumes an interaction can update the state of both agents simultaneously.
- One-way interactions:
  - Information can flow from sender to receiver, but not vice-versa.

Variants

- Is sender aware that it has sent a message?
- Does the information flow instantaneously?
  - Immediate transmission: message delivered instantly
  - Immediate observation: receiver sees sender's current state
  - Delayed transmission: unpredictable delay in delivery
  - Delayed observation: receiver sees an old state of sender
- Can incoming message be queued?

Overview

Exact characerizations exist:

- Delayed observation: can count up to 2.
- Immediate observation: can count up to any constant.
- Immediate and delayed transmission: characterization exists
  - strictly stronger than observation (can compute mods)
  - strictly weaker than two-way (cannot compute majority)
- Queued transmission is equivalent to two-way interactions.

Plan

Our question

- Can one say that all protocols are games?
- What is the power of protocols that correspond to games?

Basic of Game theory

- Two players games: 1 and 2,
  with a finite set of *pure strategies*, $Strat(1)$ and $Strat(2)$.
- Denote by $A_{i,j}$ (respectively: $B_{i,j}$) the score for player 1 (resp. 2) when

  1 uses strategy $i \in Strat(1)$
  and 2 uses strategy $j \in Strat(2)$.
- The game is termed *symmetric* if A is the transpose of B.
- Famous prisonner's dilemma:

Player \ Opponent	🟢	🔴
🟢	2	−1
🔴	1	0

where $Strat(1) = Strat(2) = \{🟢, 🔴\}$.

Best response

- A strategy $x \in Strat(1)$ is said to be a best response to strategy $y \in Strat(II)$, denoted by $x \in BR(y)$ if

$$A_{z,y} \leq A_{x,y} \tag{1}$$

  for all strategies $z \in Strat(I)$.

- In the prisonner's dilemma,

$$\text{BR}(\bullet)=\text{BR}(\bullet)=\bullet,$$

  $\bullet$is the best rational choice, but $\bullet$ would be the better social choice.

- We write $\mathbf{x} \in BR_{\neq \mathbf{x}'}(\mathbf{y})$ for

$$A_{z,y} \leq A_{x,y} \tag{2}$$

  for all strategy $\mathbf{z} \in Strat(\text{I}), \mathbf{z} \neq \mathbf{x}'$.

Turning a Game into a Dynamic: Pavlovian's behavior

Assume a two-player game is given. Let Δ be some threshold.

- The protocol associated to the game is a population protocol whose set of states is $Q = Strat(1) = Strat(2)$ and whose transition rules δ are given as follows:

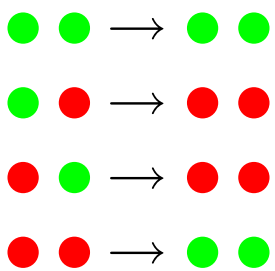
$$q_1, q_2 \rightarrow q_1', q_2'$$

  where
  - $q_1' = q_1$ if $A_{q_1,q_2} \geq \Delta$, otherwise $q_1' \in BR_{\neq q_1}(q_2)$ $(A_{q_1,q_2} < \Delta)$
  - $q_2' = q_2$ if $B_{q_1,q_2} \geq \Delta$, otherwise $q_2' \in BR_{\neq q_1}(q_2)$ $(B_{q_1,q_2} < \Delta)$
- A population protocol is Pavlovian if it can be obtained from a game as above.

Example: The Prisonner's Dilemma

Player \ Opponent	C	D
C	2	-1
D	1	0

Taking $\Delta = 0$,



- Several studies of this dynamic over various graphs: see e.g. [Dyer et al. 02], [Fribourg et al. 04].
- Somehow, our question is: can any dynamic be termed "a game", or "pavlovian".

First observation

- Write any rule of the protocol

$$q_1 q_2 \to \delta_1(q_1, q_2)\delta_1(q_2, q_1)$$

- Consider a 3-states population protocol with set of states $Q = \{$●,●,●$\}$ and a joint transition function δ such that

$$\delta_1(●,●) =●,\ \delta_1(●,●) =●,\ \delta_1(●,●) =●.$$

- One can not find a matrix of a game that would lead to this dynamic.

- Corollary: Not all protocols are Pavlovian.

Symmetric Population Protocols

Second observation

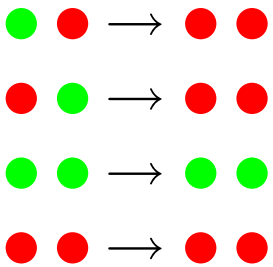
We say that a population protocol is symmetric if, whenever $q_1, q_2 \to q_1', q_2'$ in the program, one has also $q_2, q_1 \to q_2', q_1'$.

Theorem

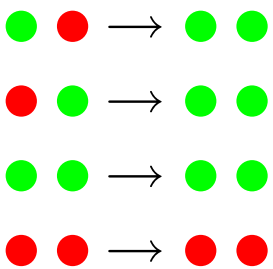
Any symmetric deterministic 2-states population protocol is Pavlovian.

Basic Pavlovian Protocols

- *OR* is computed by 2-state protocol:

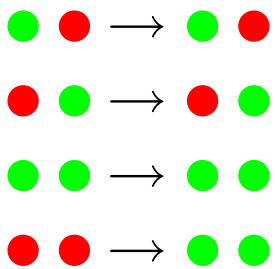


- *AND* is computed by 2-state protocol:



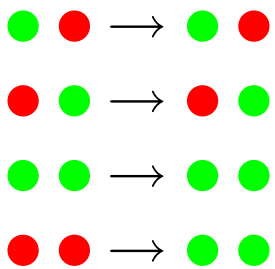
Remark

XOR is not computed by 2-state protocol:



Remark

XOR is not computed by 2-state protocol:

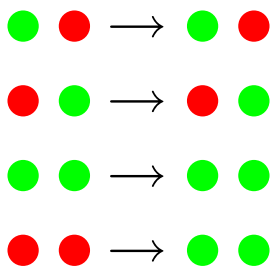


The answer is not eventually known by all the agents.

- eventually, all agents will be in state 0, if the XOR of input bits is 0,
- eventually only one agent will be in state 1, if the XOR of input bits is 1.

Remark

XOR is not computed by 2-state protocol:

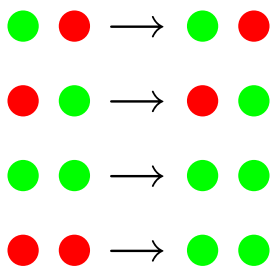


The answer is not eventually known by all the agents.

- eventually, all agents will be in state 0, if the XOR of input bits is 0,
- eventually only one agent will be in state 1, if the XOR of input bits is 1.

Remark

XOR is not computed by 2-state protocol:



The answer is not eventually known by all the agents.

- eventually, all agents will be in state 0, if the XOR of input bits is 0,
- eventually only one agent will be in state 1, if the XOR of input bits is 1.

Electing a Leader

Classical solution: $\bullet\bullet \to \bullet\bullet$ is not symmetric.

Proposition

A symetric following Pavlovian protocol solves the leader election problem, as soon as the size of population is ≥ 3.

Proof: Initially, all agents in same state L_1.

Let consider the following population protocol :

$L_1\ L_1 \to L_2\ L_2$ $\quad$ $L_2\ L_1 \to L_1\ N$ $\quad$ $N\ L_1 \to L_2\ L_1$

$L_2\ L_2 \to L_1\ L_1$ $\quad$ $N\ N \to N\ N$ $\quad$ $L_2\ N \to N\ L_1$

Electing a Leader

Classical solution: $\bullet\bullet \to \bullet\bullet$ is not symmetric.

Proposition

A symetric following Pavlovian protocol solves the leader election problem, as soon as the size of population is ≥ 3.

Proof: Initially, all agents in same state L_1.
Let consider the following population protocol :

$L_1\ L_1 \to L_2\ L_2$ $\quad$ $L_2\ L_1 \to L_1\ N$ $\quad$ $N\ L_1 \to L_2\ L_1$

$L_2\ L_2 \to L_1\ L_1$ $\quad$ $N\ N \to N\ N$ $\quad$ $L_2\ N \to N\ L_1$

Correctness

- The number of agents in states L_1 or L_2 decreases
- Eventually, exactly one agent is in a special leader state (L_1 or L_2).

Indeed, ultimately there will be exactly one leader, that is one agent in state L_1 or L_2.

Electing a Leader

Classical solution: $\bullet\ \bullet \to \bullet\ \bullet$ is not symmetric.

Proposition

A symetric following Pavlovian protocol solves the leader election problem, as soon as the size of population is ≥ 3.

Proof: Initially, all agents in same state L_1.

Let consider the following population protocol :

$L_1\ L_1 \to L_2\ L_2$ $\quad$ $L_2\ L_1 \to L_1\ N$ $\quad$ $N\ L_1 \to L_2\ L_1$

$L_2\ L_2 \to L_1\ L_1$ $\quad$ $N\ N \to N\ N$ $\quad$ $L_2\ N \to N\ L_1$

Taking $\Delta = 0$, this corresponds to matrix

Player \ Opponent	L_1	L_2	N
L_1	-3	1	-1
L_2	-1	-1	-1
N	-2	-1	1

Third observation

Assume that $\Delta = 0$

We can always assume that the matrix corresponding to a Pavlovian population protocol has three types of columns

- columns containing only 0s
- columns containing only one 1s, and some 0s, -1s
- columns containing only one -1s, one -2, and some -3s

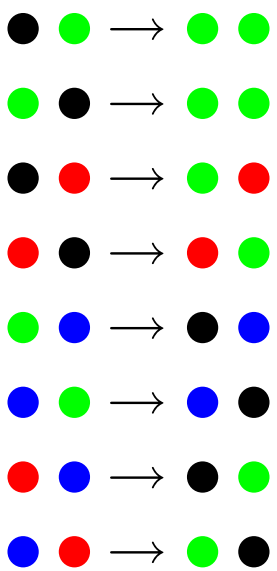
		Opponent		
		L_1	L_2	N
Player	L_1	-3	1	-1
	L_2	-1	-1	-1
	N	-2	-1	1

Majority ($\#\bullet \geq \#\bullet$)

Proposition

The majority problem (given some population of $\bullet$ and $\bullet$, determine whether there are more $\bullet$ than $\bullet$) can be solved by a Pavlovian population protocol.

Proof:



Majority (#● ≥ # ●)

Proposition

The majority problem (given some population of ● and ●, determine whether there are more ● than ●) can be solved by a Pavlovian population protocol.

Proof:

This corresponds to the following matrix.

		Opponent ●	Opponent ●	Opponent ●	Opponent ●
Player	●(N)	1	−1	−1	1
	●(Y)	0	1	1	−1
	●	0	0	0	−1
	●	0	0	−1	0

How to Compute $[x.\sigma \geq 3]$

Proposition

There is a symetric Pavlovian protocol that computes the threshold predicate $[x.\sigma \geq 3]$, which is true when there are at least 3 occurrences of input symbol σ in the input x.

Proof:

- Step 1: find a population protocol
- Step 2: show that it is pavlovian (give the matrix)

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$00 \rightarrow 00$
$01 \rightarrow 01$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$00 \to 00$
$01 \to 01$
$11 \to 2_1 2_1$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$
\begin{aligned}
00 &\to 00\\
01 &\to 01\\
11 &\to 2_1 2_1\\
2_1 2_1 &\to 2_2 2_2\\
2_2 2_2 &\to 2_1 2_1
\end{aligned}
$$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$
\begin{array}{ll}
00 \to 00 & 02_2 \to 2_1 0 \\
01 \to 01 & 02_1 \to 2_2 0 \\
11 \to 2_1 2_1 & \\
2_1 2_1 \to 2_2 2_2 & \\
2_2 2_2 \to 2_1 2_1 &
\end{array}
$$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$
\begin{array}{ll}
00 \to 00 & 02_2 \to 2_1 0 \\
01 \to 01 & 02_1 \to 2_2 0 \\
11 \to 2_1 2_1 & \\
2_1 2_1 \to 2_2 2_2 & 12_1 \to 2_1 2_2 \\
2_2 2_2 \to 2_1 2_1 & 12_2 \to 2_2 2_1
\end{array}
$$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$
\begin{array}{ll}
00 \to 00 & 02_2 \to 2_1 0 \\
01 \to 01 & 02_1 \to 2_2 0 \\
11 \to 2_1 2_1 & 2_2 2_1 \to YX \\
2_1 2_1 \to 2_2 2_2 & 12_1 \to 2_1 2_2 \\
2_2 2_2 \to 2_1 2_1 & 12_2 \to 2_2 2_1
\end{array}
$$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$
\begin{array}{lll}
00 \to 00 & 02_2 \to 2_1 0 & 0X \to 0X \\
01 \to 01 & 02_1 \to 2_2 0 & 0Y \to 0Y \\
11 \to 2_1 2_1 & 2_2 2_1 \to YX & 1X \to 3X \\
2_1 2_1 \to 2_2 2_2 & 12_1 \to 2_1 2_2 & 1Y \to 3Y \\
2_2 2_2 \to 2_1 2_1 & 12_2 \to 2_2 2_1 & 13 \to 33
\end{array}
$$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$
\begin{array}{lll}
00 \to 00 & 02_2 \to 2_1 0 & 0X \to 0X \\
01 \to 01 & 02_1 \to 2_2 0 & 0Y \to 0Y \\
11 \to 2_1 2_1 & 2_2 2_1 \to YX & 1X \to 3X \\
2_1 2_1 \to 2_2 2_2 & 12_1 \to 2_1 2_2 & 1Y \to 3Y \\
2_2 2_2 \to 2_1 2_1 & 12_2 \to 2_2 2_1 & 13 \to 33
\end{array}
$$

$$
\begin{array}{l}
2_1 X \to 32_2 \\
2_1 Y \to 32_2 \\
2_2 X \to 32_1 \\
2_2 Y \to 32_1
\end{array}
$$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$00 \to 00 \qquad 02_2 \to 2_1 0 \qquad 0X \to 0X$$
$$01 \to 01 \qquad 02_1 \to 2_2 0 \qquad 0Y \to 0Y$$
$$11 \to 2_1 2_1 \qquad 2_2 2_1 \to YX \qquad 1X \to 3X$$
$$2_1 2_1 \to 2_2 2_2 \qquad 12_1 \to 2_1 2_2 \qquad 1Y \to 3Y$$
$$2_2 2_2 \to 2_1 2_1 \qquad 12_2 \to 2_2 2_1 \qquad 13 \to 33$$

$$2_1 X \to 32_2 \qquad 03 \to 33$$
$$2_1 Y \to 32_2 \qquad YY \to 33$$
$$2_2 X \to 32_1 \qquad 33 \to 33$$
$$2_2 Y \to 32_1 \qquad XY \to XY$$
$$X3 \to 33$$
$$XX \to 33$$

The population protocol

- $Q = \{0, 1, 2_1, 2_2, X, Y, 3\}$.
- $\Sigma = \{x, y\}$.
- $\iota(x) = 1$, $\iota(y) = 0$.
- $\omega = 1_{\{3\}}$.
- Its transition function can be written as follows:

$$
\begin{array}{lll}
00 \to 00 & 02_2 \to 2_1 0 & 0X \to 0X \\
01 \to 01 & 02_1 \to 2_2 0 & 0Y \to 0Y \\
11 \to 2_1 2_1 & 2_2 2_1 \to YX & 1X \to 3X \\
2_1 2_1 \to 2_2 2_2 & 12_1 \to 2_1 2_2 & 1Y \to 3Y \\
2_2 2_2 \to 2_1 2_1 & 12_2 \to 2_2 2_1 & 13 \to 33
\end{array}
$$

$$
\begin{array}{ll}
2_1 X \to 32_2 & 03 \to 33 \\
2_1 Y \to 32_2 & YY \to 33 \\
2_2 X \to 32_1 & 33 \to 33 \\
2_2 Y \to 32_1 & XY \to XY \\
2_2 3 \to 2_2 2_1 & X3 \to 33 \\
2_1 3 \to 2_1 2_2 & XX \to 33
\end{array}
$$

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

$w2_1 \rightarrow 2_2u$ with $w \neq 2_2$ and $2_22_1 \rightarrow 2_2y$
$w2_2 \rightarrow 2_1u$ with $w \neq 2_2$ and $2_12_2 \rightarrow 2_1y$

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

$$* \, 3 \to 3y,\ 0X \to 0X,\ YX \to YX,$$
$$wX \to 3u \text{ with } w \neq 0, Y$$

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

$w2_1 \rightarrow 2_2u$ with $w \neq 2_2$ and $2_22_1 \rightarrow 2_2y$
$w2_2 \rightarrow 2_1u$ with $w \neq 2_2$ and $2_12_2 \rightarrow 2_1y$

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

$* \; 3 \to 3y,\; 0X \to 0X,\; YX \to YX,$
$wX \to 3u$ with $w \neq 0, Y$

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

$w2_1 \rightarrow 2_2u$ with $w \neq 2_2$ and $2_22_1 \rightarrow 2_2y$
$w2_2 \rightarrow 2_1u$ with $w \neq 2_2$ and $2_12_2 \rightarrow 2_1y$

Its payoff matrix

	0	1	2_1	2_2	X	Y	3
0	1	0	-3	-3	0	0	-1
1	0	-1	-3	-3	-1	-1	-1
2_1	-1	1	-3	-1	-1	-1	0
2_2	-1	0	-1	-3	-1	-1	0
X	0	0	-2	-3	-1	0	-1
Y	0	0	-3	-2	0	-1	-1
3	0	0	-3	-3	1	1	1

Remark:

$* 3 \rightarrow 3y$, $0X \rightarrow 0X$, $YX \rightarrow YX$,
$wX \rightarrow 3u$ with $w \neq 0, Y$

Summary

Theorem

There is a symetric Pavlovian protocol that computes the threshold predicate $[x.\sigma \geq 2^k]$, with the integer $k \geq 2$ which is true when there are at least 2^k occurrences of input symbol σ in the input x.

Conjecture

There does not exist a symetric Pavlovian protocol that computes the predicate $x \equiv 0 \pmod 2$.

Asymmetric Population Protocols

Asymmetric Population Protocol

Any rule of the protocol is such that

$$q_1 q_2 \to \delta_1(q_1, q_2)\delta_2(q_2, q_1)$$

There are one-way interactions:

Information flows from sender to receiver

For example : leader election with $(\Delta = 0)$ (● ● → ● ●)

Sender

	Opponent ●	Opponent ●
●	−1	0
●	1	0

Receiver

	Opponent ●	Opponent ●
●	0	0
●	0	0

Principle:

Sum of agents' values is invariant.

$$xy \longrightarrow xy \text{ if } x > y$$
$$xy \longrightarrow y+1 \;\; x-1 \text{ if } x \leq y$$

Addition

$xy \rightarrow xy$ if $x > y$ $\qquad$ $xy \rightarrow y+1\ x-1$ if $x \leq y$

SENDER	A	0	1	...	k-1	k	k+1	...	Max
	0	-1	-1	-1	-1	-1	-1	-1	0
	1	1	-1	-1	-1	-1	-1	-1	0
	...	0	1	-1	-1	-1	-1	-1	0
	k-1	0	0	1	-1	-1	-1	-1	0
	k	0	0	0	1	-1	-1	-1	0
	k+1	0	0	0	0	1	-1	-1	0
	...	0	0	0	0	0	1	-1	0
	Max	0	0	0	0	0	-1	1	0

RECEIVER	D	0	1	...	k-1	k	k+1	...	Max
	0	-1	1	0	0	0	0	0	0
	1	-1	-1	1	0	0	0	0	0
	...	-1	-1	-1	1	0	0	0	0
	k-1	-1	-1	-1	-1	1	0	0	0
	k	-1	-1	-1	-1	-1	1	0	0
	k+1	-1	-1	-1	-1	-1	-1	1	0
	...	-1	-1	-1	-1	-1	-1	-1	0
	Max	0	0	0	0	0	0	0	0

A predicate $\sum_{i=1}^{k} x_i \geq Max$

Principle:

Addition + broadcasting

$$x \;\; y \longrightarrow x \;\; y \text{ if } x > y \text{ and } y \neq Max$$
$$x \;\; y \longrightarrow y+1 \;\; x-1 \text{ if } x \leq y \text{ and } y \neq Max$$
$$x \;\; Max \longrightarrow Max \;\; Max$$

A predicate $\sum_{i=1}^{k} x_i \geq Max$

RECEIVER	D	0	1	...	k-1	k	k+1	...	Max
	0	-1	1	0	0	0	0	0	0
	1	-1	-1	1	0	0	0	0	0
	...	-1	-1	-1	1	0	0	0	0
	k-1	-1	-1	-1	-1	1	0	0	0
	k	-1	-1	-1	-1	-1	1	0	0
	k+1	-1	-1	-1	-1	-1	-1	1	0
	...	-1	-1	-1	-1	-1	-1	-1	0
	Max	0	0	0	0	0	0	0	0

A predicate $\sum_{i=1}^{k} x_i \geq Max$

SENDER

A	0	1	...	k-1	k	k+1	...	Max
0	-1	-1	-1	-1	-1	-1	-1	**-1**
1	1	-1	-1	-1	-1	-1	-1	**-1**
...	0	1	-1	-1	-1	-1	-1	**-1**
k-1	0	0	1	-1	-1	-1	-1	**-1**
k	0	0	0	1	-1	-1	-1	**-1**
k+1	0	0	0	0	1	-1	-1	**-1**
...	0	0	0	0	0	1	-1	**-1**
Max	0	0	0	0	0	-1	1	**1**

RECEIVER

D	0	1	...	k-1	k	k+1	...	Max
0	-1	1	0	0	0	0	0	0
1	-1	-1	1	0	0	0	0	0
...	-1	-1	-1	1	0	0	0	0
k-1	-1	-1	-1	-1	1	0	0	0
k	-1	-1	-1	-1	-1	1	0	0
k+1	-1	-1	-1	-1	-1	-1	1	0
...	-1	-1	-1	-1	-1	-1	-1	0
Max	0	0	0	0	0	0	0	0

Computable Predicates

Theorem (2009)

A predicate is computable iff it is on the following list.

- $\sum_{i=1}^{k} c_i x_i \geq a$, *where* a, c_i*'s are integer constants*
- $\sum_{i=1}^{k} c_i x_i \equiv a \pmod{b}$ *where* a, b *and* c_i*'s are constants*

No Pavlovian protocols for boolean combinations !!!

Parallel Games

Each player/agent can play several games at the same time

One game represents one predicate to be computed

At each step, games are executed independently

Theorem (2009)

A predicate is computable with population protocol iff it is computable with Pavlovian population protocol with parallel Games.

Conclusion

Pavlovian $\subseteq$ Pavlovian Asymmetric $\subseteq$ Population Protocol

We need some technique to prove some negative results

Is-it possible to compute $x \equiv 0 \pmod 2$
for Symmetric Pavlovian protocol.

What happen when the population is large.

How important is the notion of fairness ?