Stand Up Indulgent Gathering

Quentin Bramas, Anissa Lamani, and Sébastien Tixeuil

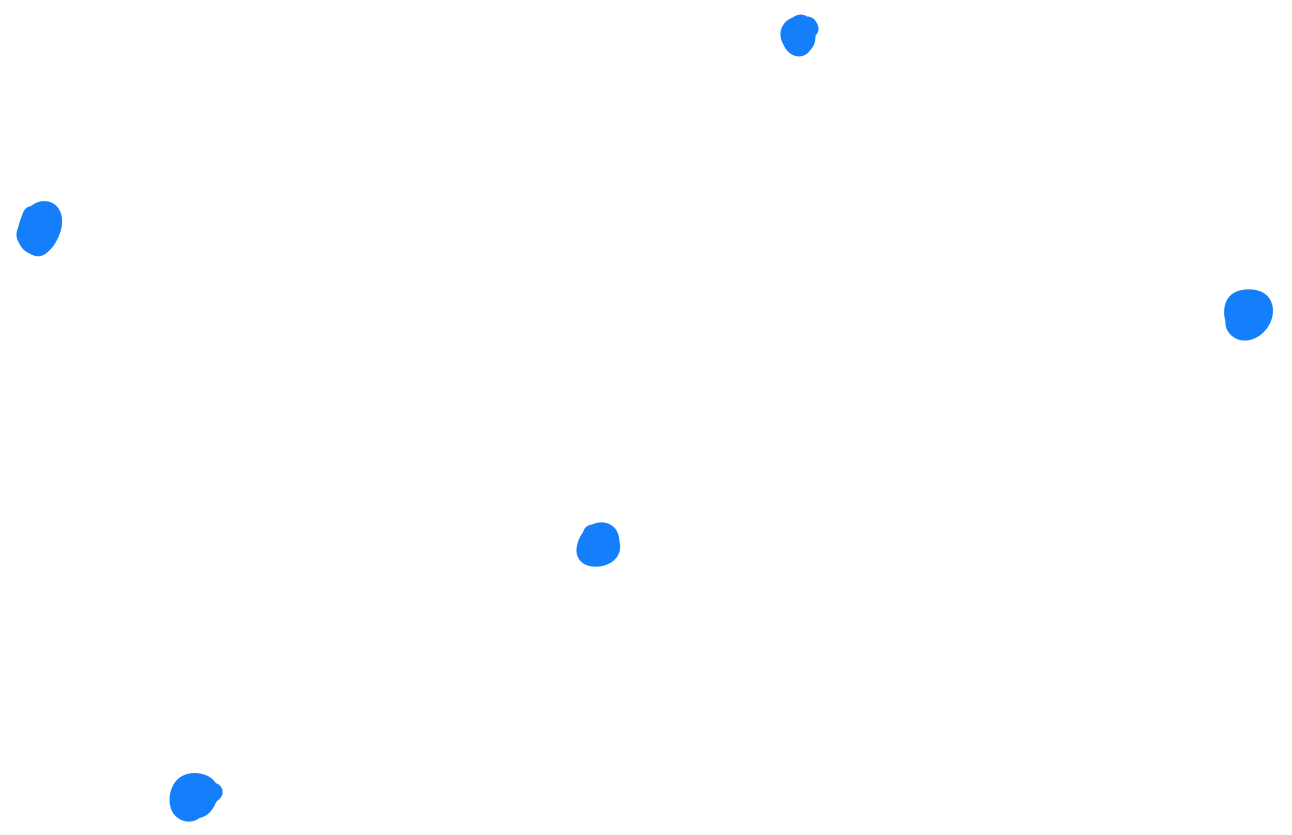
Strasbourg University, Strasbourg University, and Sorbonne University

Quentin Bramas < bramas@unistra.fr >

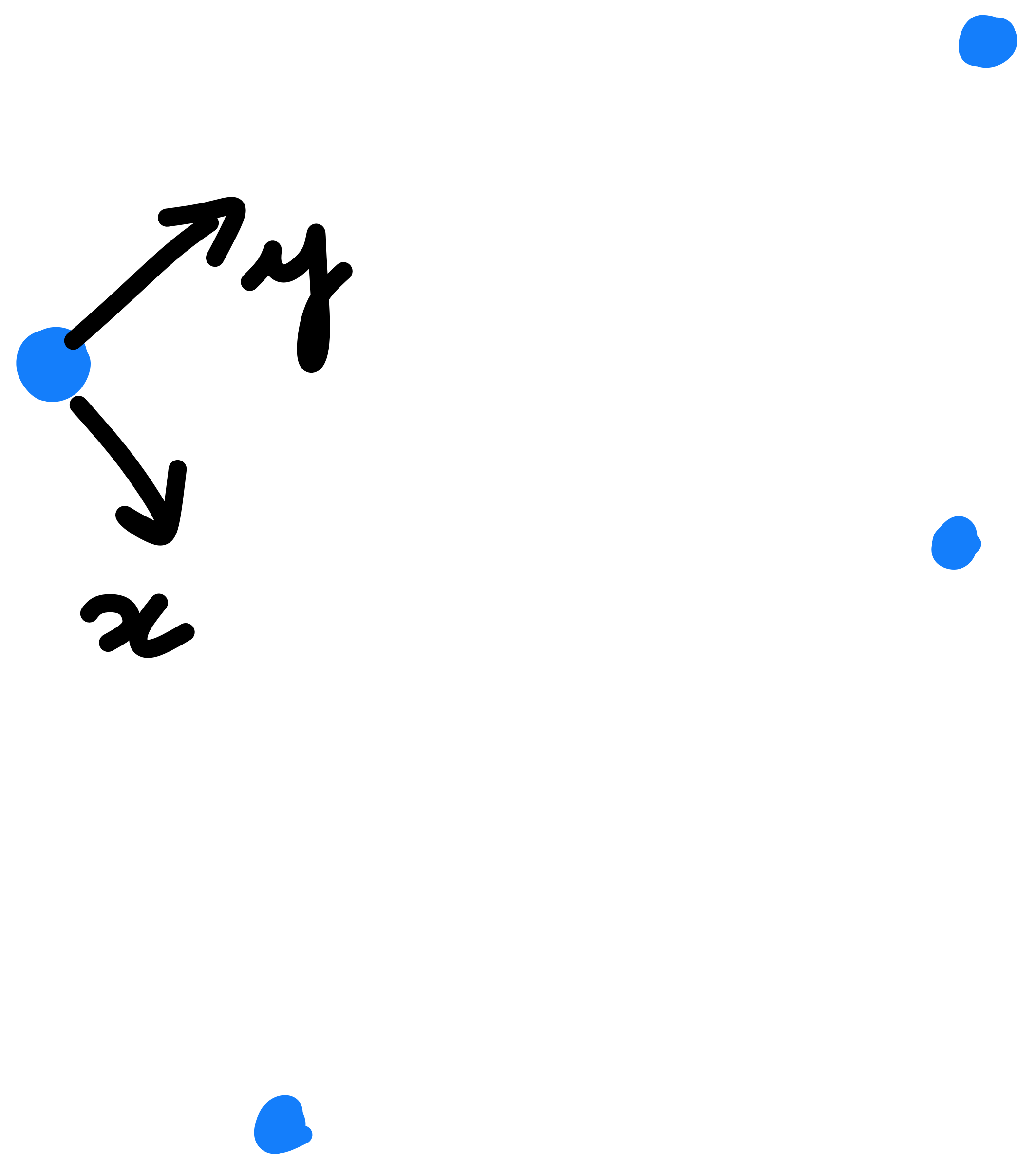
ANR ESTATE Workshop, March 16th 2022

Mobile Autonomous Robots

- Anonymous
- Uniform
- Disoriented
- Silent
- Oblivious



Execution Cycle

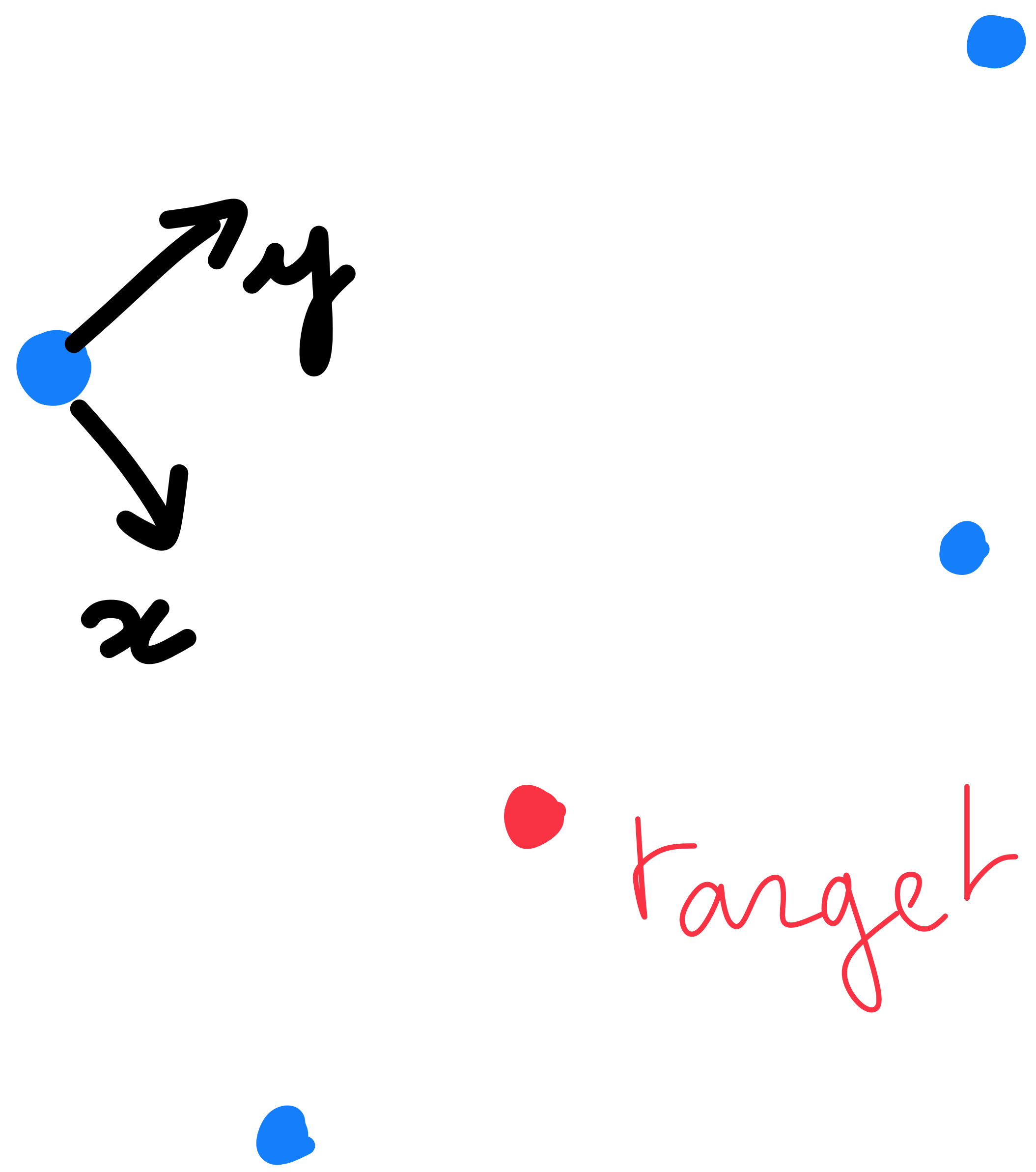


- Look
- Compute
- Move

- Fully-Synchronous
- Semi-Synchronous

Execution Cycle

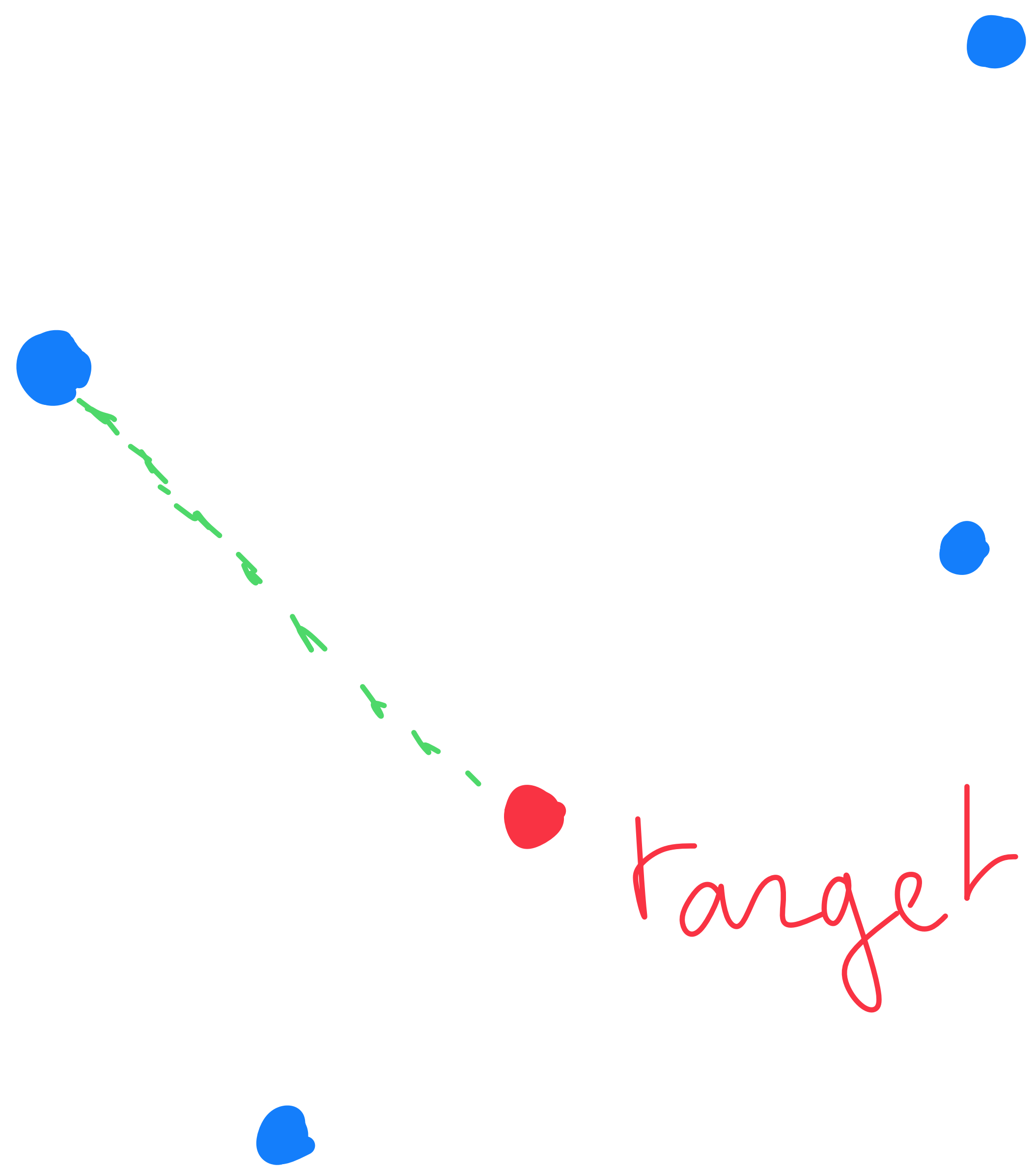
- Look
- Compute
- Move



- Fully-Synchronous
- Semi-Synchronous

Execution Cycle

- Look
- Compute
- Move



- Fully-Synchronous
- Semi-Synchronous

Execution Cycle

- Look
- Compute
- Move

- Fully-Synchronous
- Semi-Synchronous

	t = 1	t = 2	t = 3	...
Robot 1	LCM	LCM	LCM	...
Robot 2	LCM	LCM	LCM	
Robot 3	LCM	LCM	LCM	

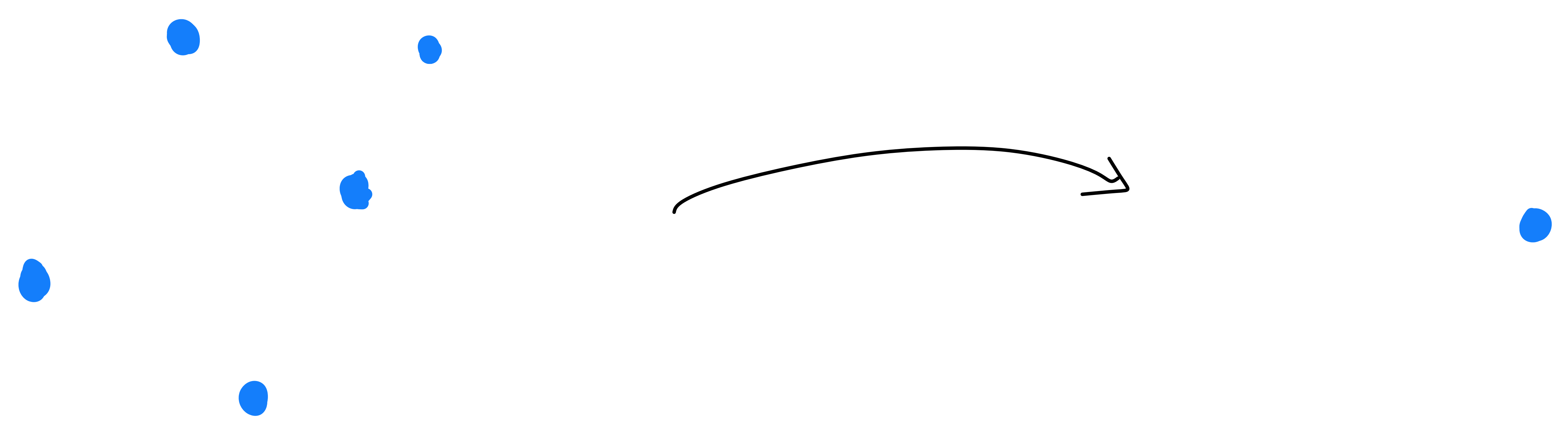
Execution Cycle

- Look
- Compute
- Move

- Fully-Synchronous
- Semi-Synchronous

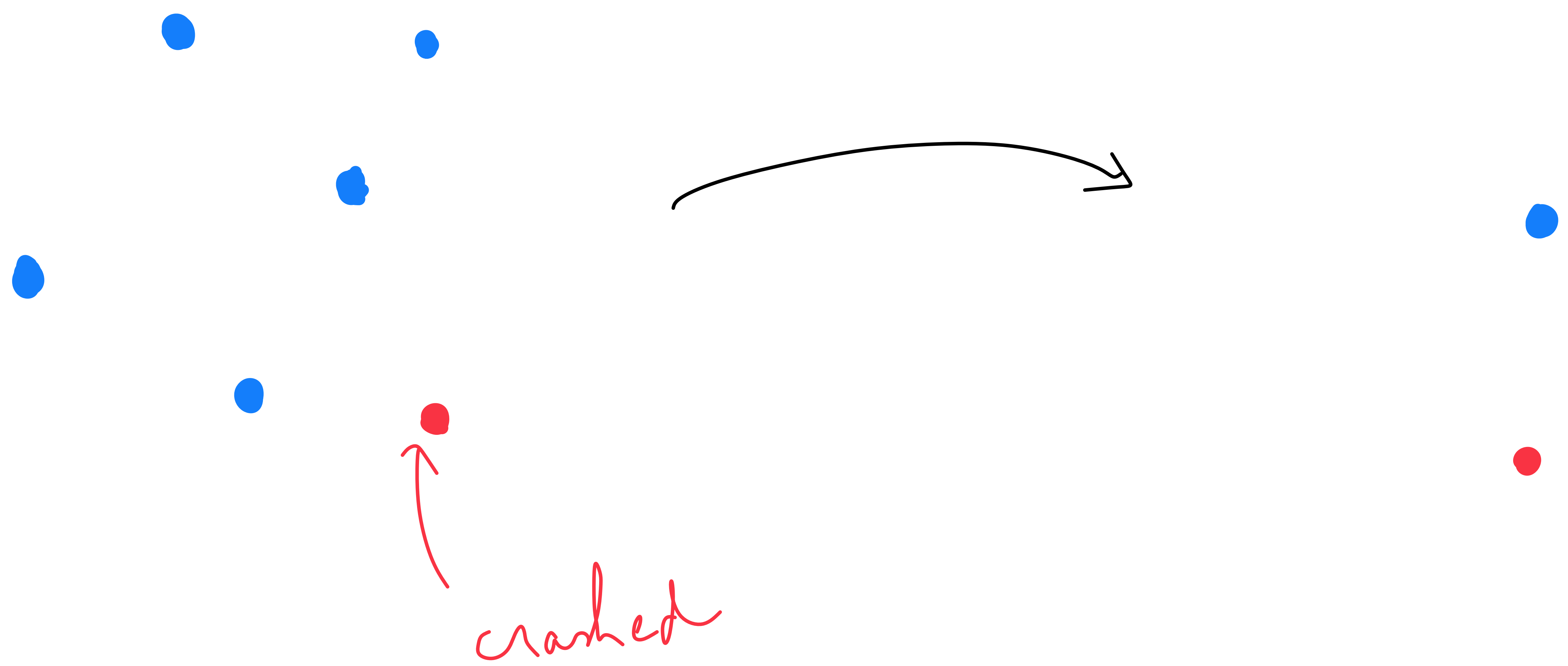
	t = 1	t = 2	t = 3	...
Robot 1	LCM		LCM	...
Robot 2		LCM		
Robot 3	LCM	LCM		

The Fundamental Problem of Gathering



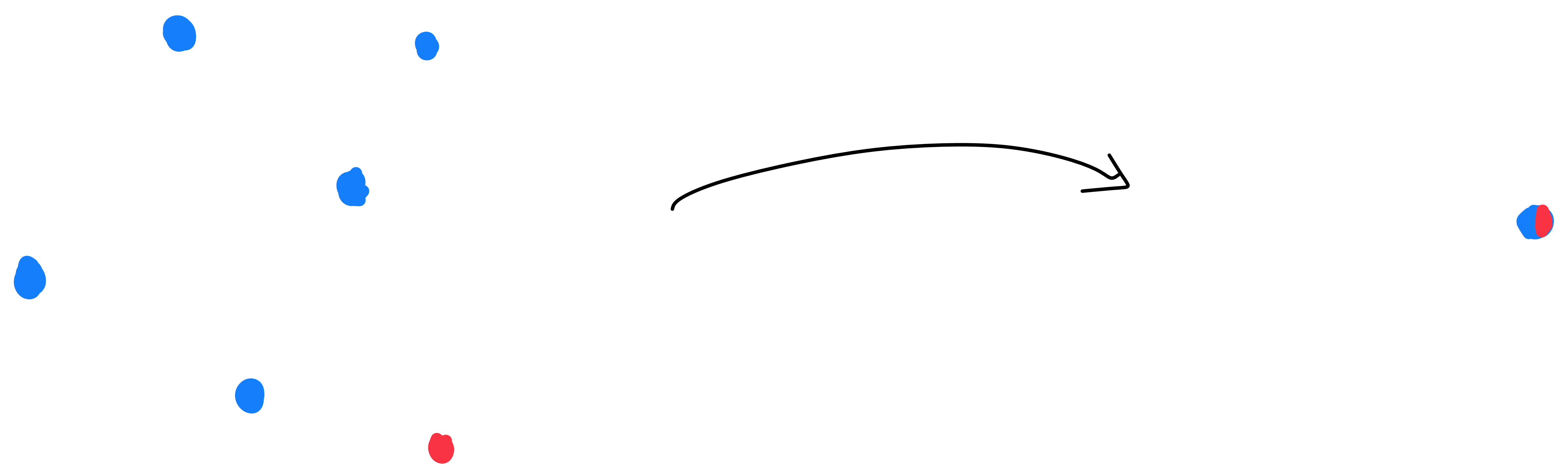
Fault-Tolerant Gathering

Weak Gathering



Stand Up Tolerant Gathering

Strong Gathering



Related Work

Related Work

Gathering:

Related Work

Gathering:

Solvable in FSYNC [SY1999]

Related Work

Gathering:

Solvable in FSYNC [SY1999]

Solvable in SSYNC with common coordinate system. [SY1999]

Related Work

Gathering:

- **Solvable** in FSYNC [SY1999]
- **Solvable** in SSYNC with common coordinate system. [SY1999]
- **Not solvable** in SSYNC without common coordinate system. [SY1999]

Related Work

Gathering:

Solvable in FSYNC [SY1999]

Solvable in SSYNC with common coordinate system. [SY1999]

Not solvable in SSYNC without common coordinate system. [SY1999]

Weak Gathering:

Related Work

Gathering:

Solvable in FSYNC [SY1999]

Solvable in SSYNC with common coordinate system. [SY1999]

Not solvable in SSYNC without common coordinate system. [SY1999]

Weak Gathering:

Solvable in SSYNC with up to n-1 crashes with multiplicity detection and non-bivalent initial configuration [Bramas&Tixeuil2015]

Related Work

Gathering:

Solvable in FSYNC [SY1999]

Solvable in SSYNC with common coordinate system. [SY1999]

Not solvable in SSYNC without common coordinate system. [SY1999]

Weak Gathering:

Solvable in SSYNC with up to n-1 crashes with multiplicity detection and non-bivalent initial configuration [Bramas&Tixeuil2015]

Stand Up Indulgent Gathering (Strong Gathering):

Related Work

Gathering:

Solvable in FSYNC [SY1999]

Solvable in SSYNC with common coordinate system. [SY1999]

Not solvable in SSYNC without common coordinate system. [SY1999]

Weak Gathering:

Solvable in SSYNC with up to n-1 crashes with multiplicity detection and non-bivalent initial configuration [Bramas&Tixeuil2015]

Stand Up Indulgent Gathering (Strong Gathering):

Unsolvable in SSYNC, even with common coordinate systems and lights with infinite # of colors [SSS2020]

Related Work

Gathering:

Solvable in FSYNC [SY1999]

Solvable in SSYNC with common coordinate system. [SY1999]

Not solvable in SSYNC without common coordinate system. [SY1999]

Weak Gathering:

Solvable in SSYNC with up to n-1 crashes with multiplicity detection and non-bivalent initial configuration [Bramas&Tixeuil2015]

Stand Up Indulgent Gathering (Strong Gathering):

Unsolvable in SSYNC, even with common coordinate systems and lights with infinite # of colors [SSS2020]

Solvable in FSYNC with two robots [SSS2020]

Related Work

Gathering:

- **Solvable** in FSYNC [SY1999]
- **Solvable** in SSYNC with common coordinate system. [SY1999]
- **Not solvable** in SSYNC without common coordinate system. [SY1999]

Weak Gathering:

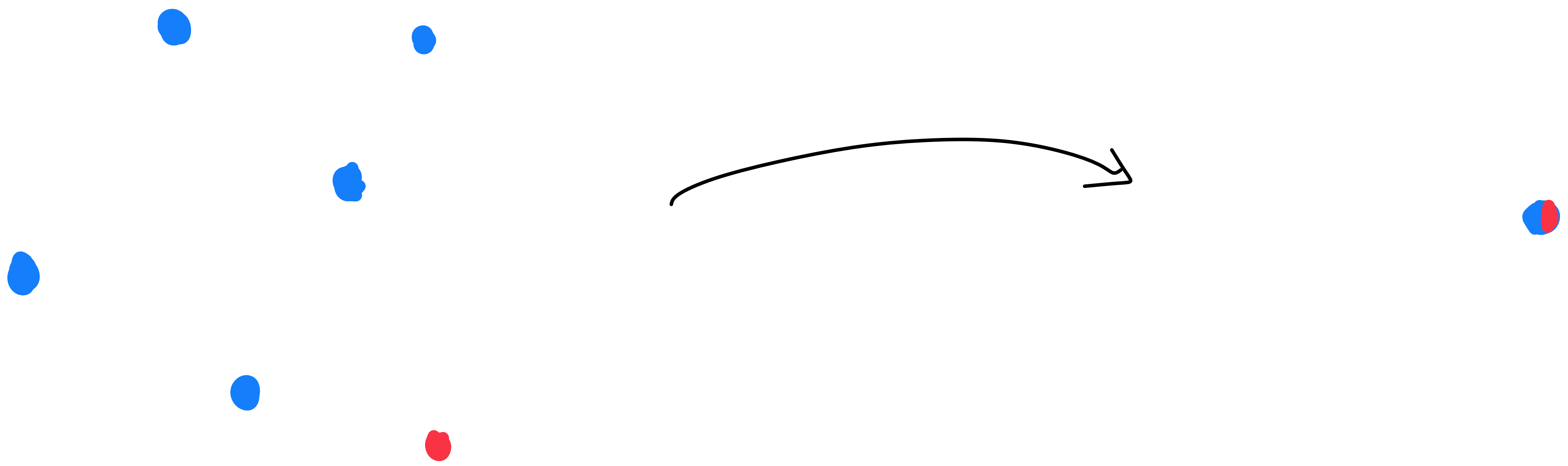
- **Solvable** in SSYNC with up to n-1 crashes with multiplicity detection and non-bivalent initial configuration [Bramas&Tixeuil2015]

Stand Up Indulgent Gathering (Strong Gathering):

- **Unsolvable** in SSYNC, even with common coordinate systems and lights with infinite # of colors [SSS2020]
- **Solvable** in FSYNC with two robots [SSS2020]
- **Solvable** in FSYNC with n robots [ALGOSENSOR2021]

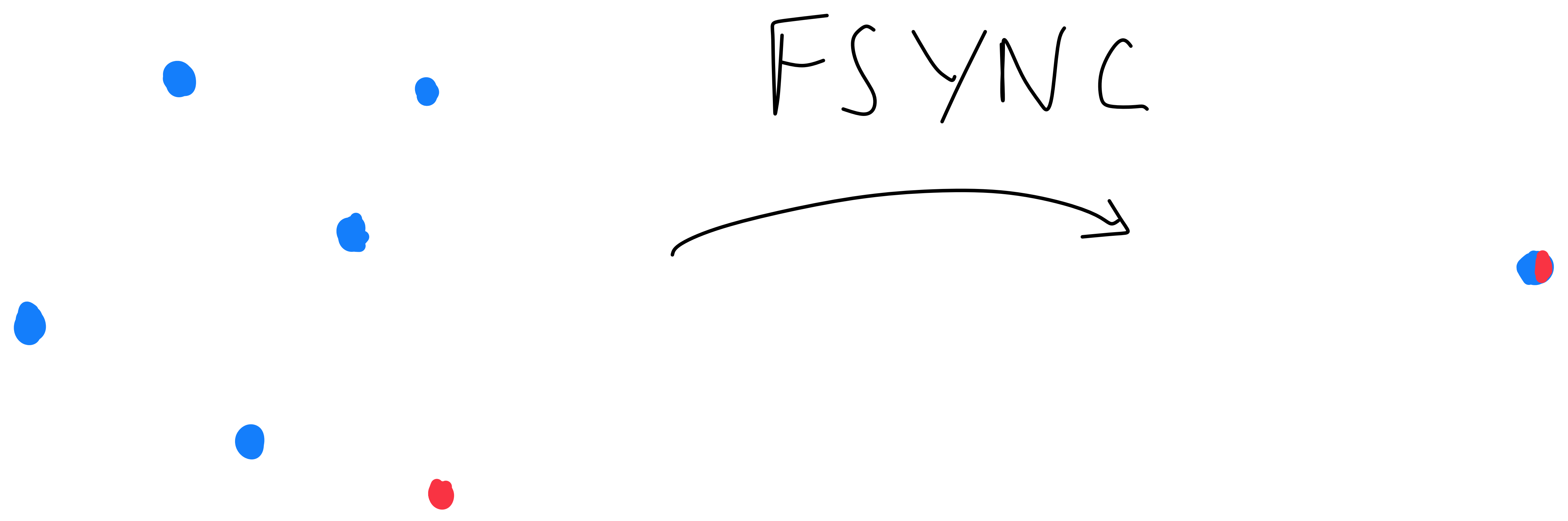
Stand Up Tolerant Gathering

Strong Gathering



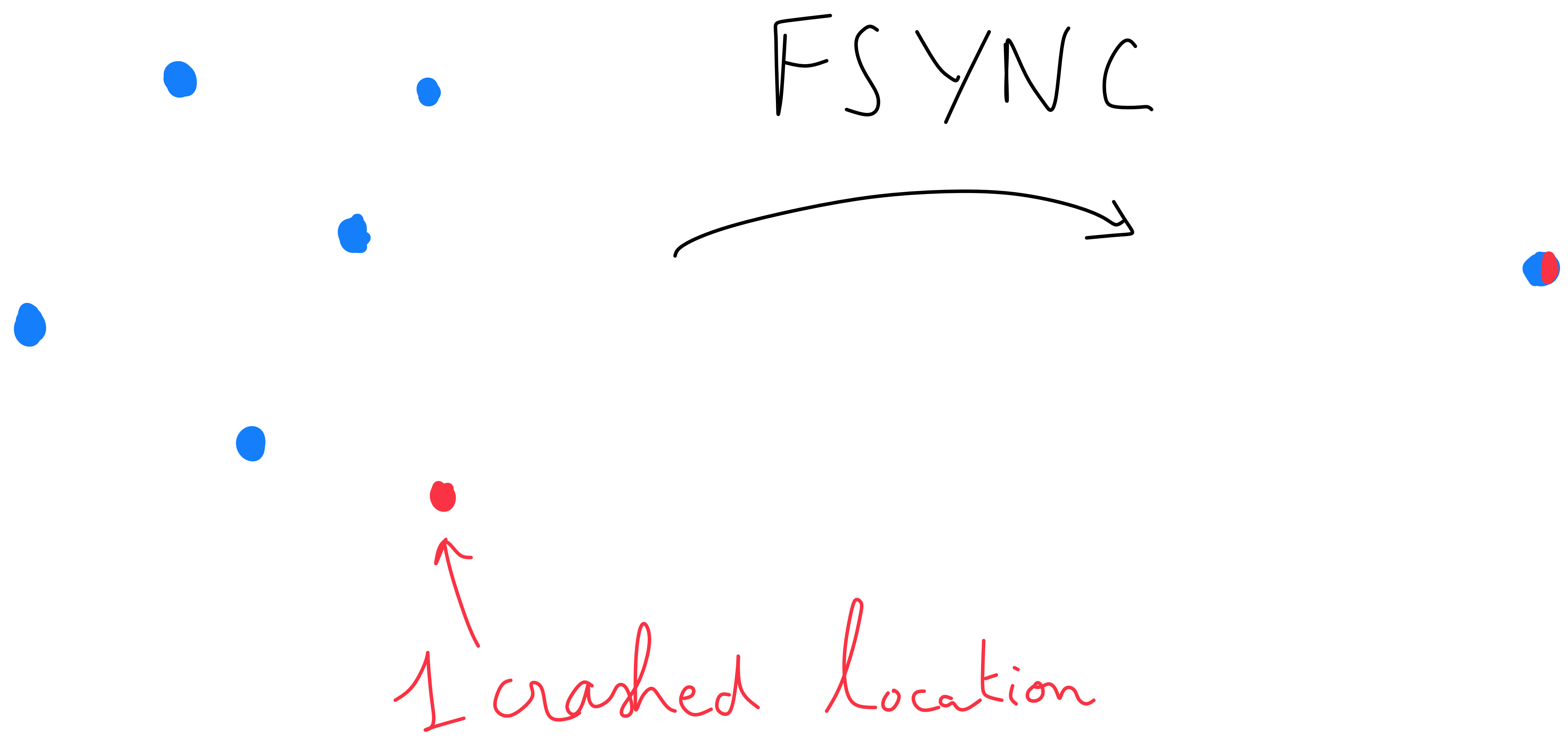
Stand Up Tolerant Gathering

Strong Gathering



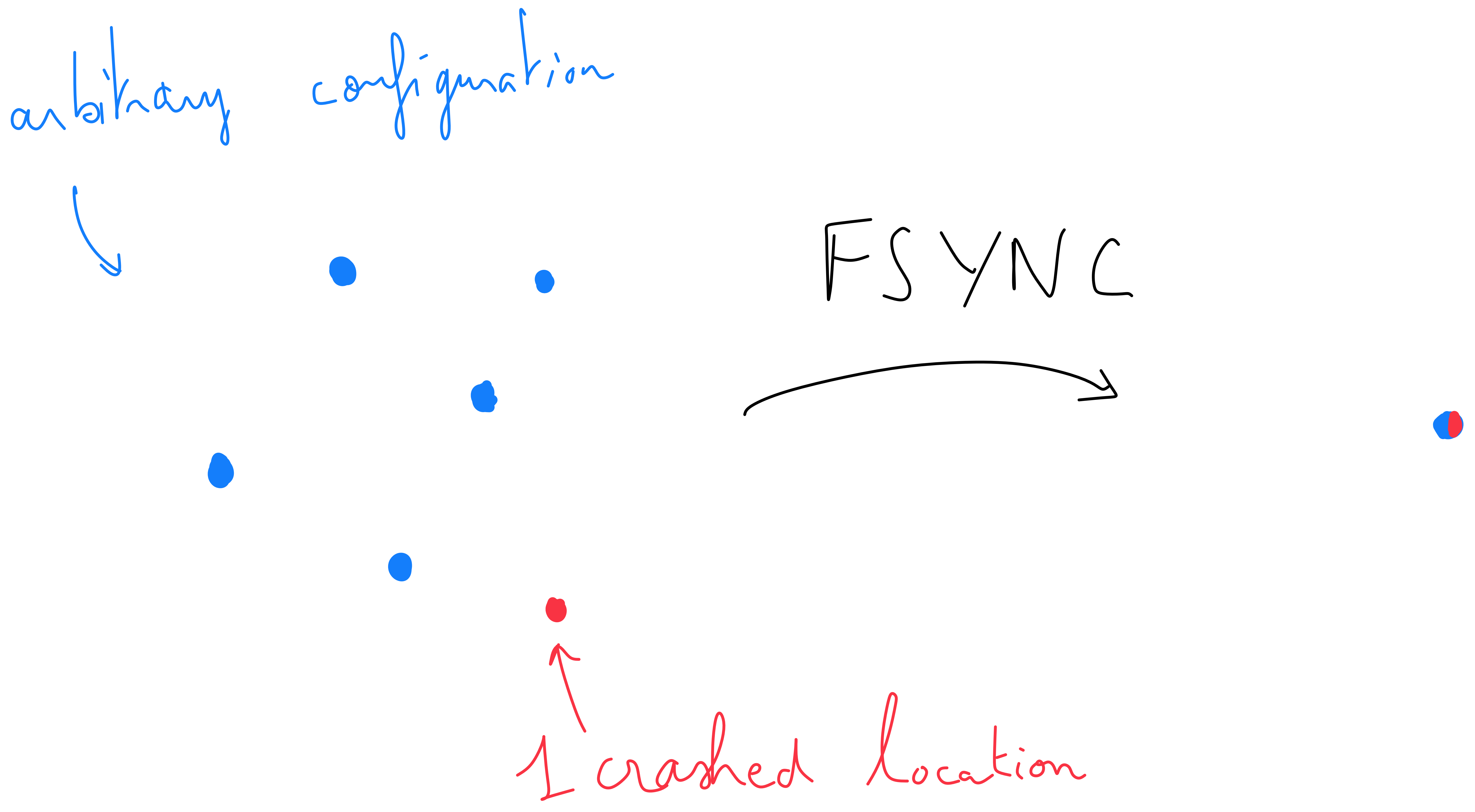
Stand Up Tolerant Gathering

Strong Gathering



Stand Up Tolerant Gathering

Strong Gathering



Stand Up Tolerant Gathering

Strong Gathering

arbitrary configuration

may contain multiplicity (but no detection)

1 crashed location

Impossibility Result

With 2 robots in SSYNC [SSS2020]

Impossibility result

[SSS2020]

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Impossibility result

[SSS2020]

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Lemma: *in an execution, if only one robot r is activated, then there is a round when r is dictated to move to the other robot.*

Impossibility result

[SSS2020]

Let A be an algorithm that solves the SUIR problem with lights
(infinite memory and communication capabilities)

Lemma: *in an execution, if only one robot r is activated, then there is a round when r is dictated to move to the other robot.*

Proof: Indeed, if the other robot is crashed, we know that in finite number of rounds r moves to the other robot.

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.

Impossibility result

[SSS2020]

Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.

Impossibility result

[SSS2020]

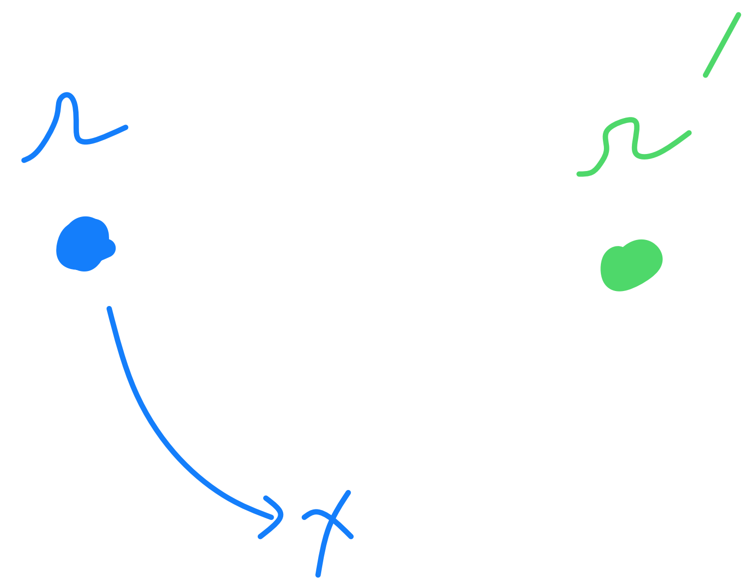
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

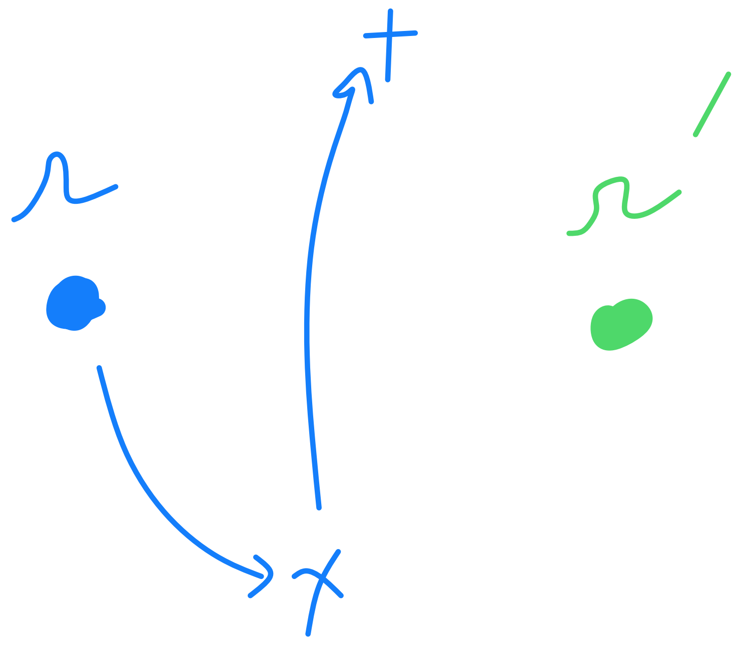
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

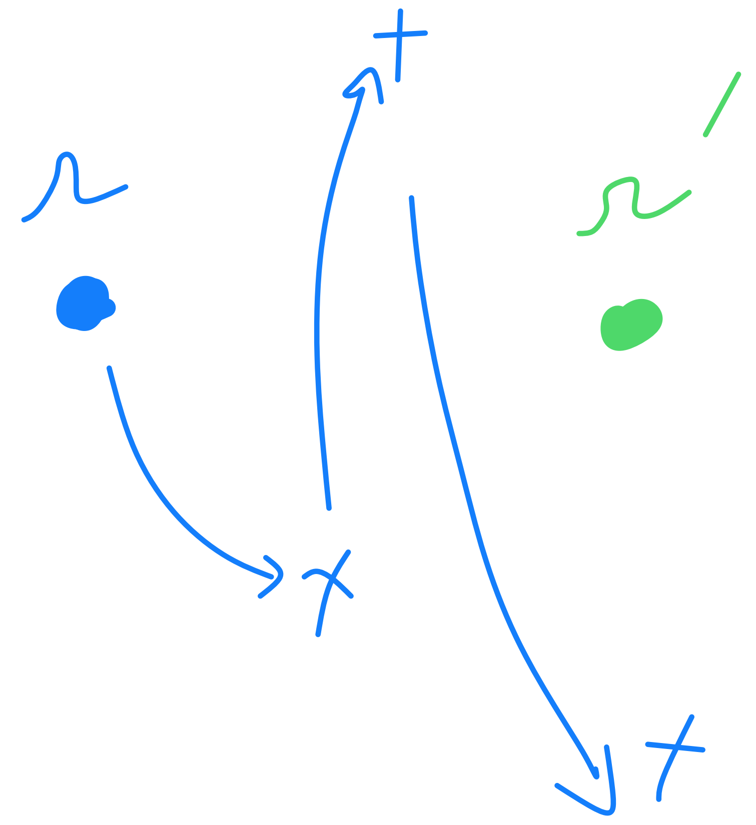
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

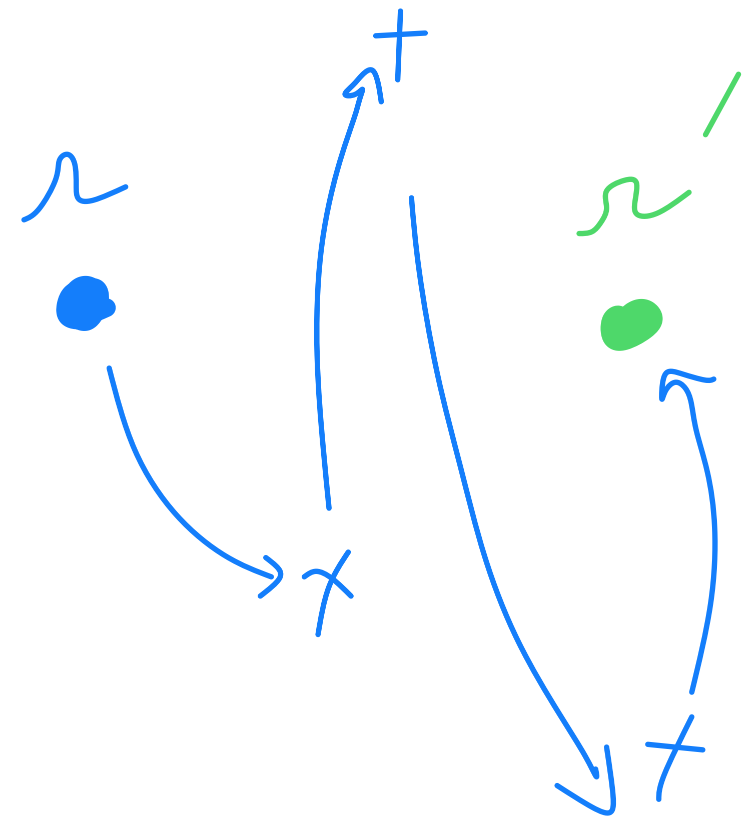
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

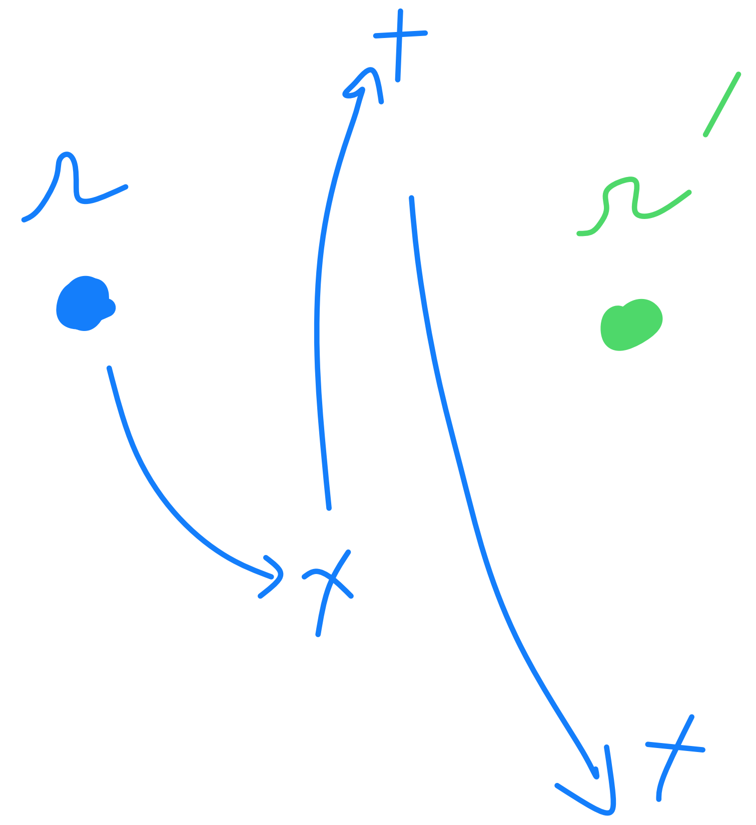
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

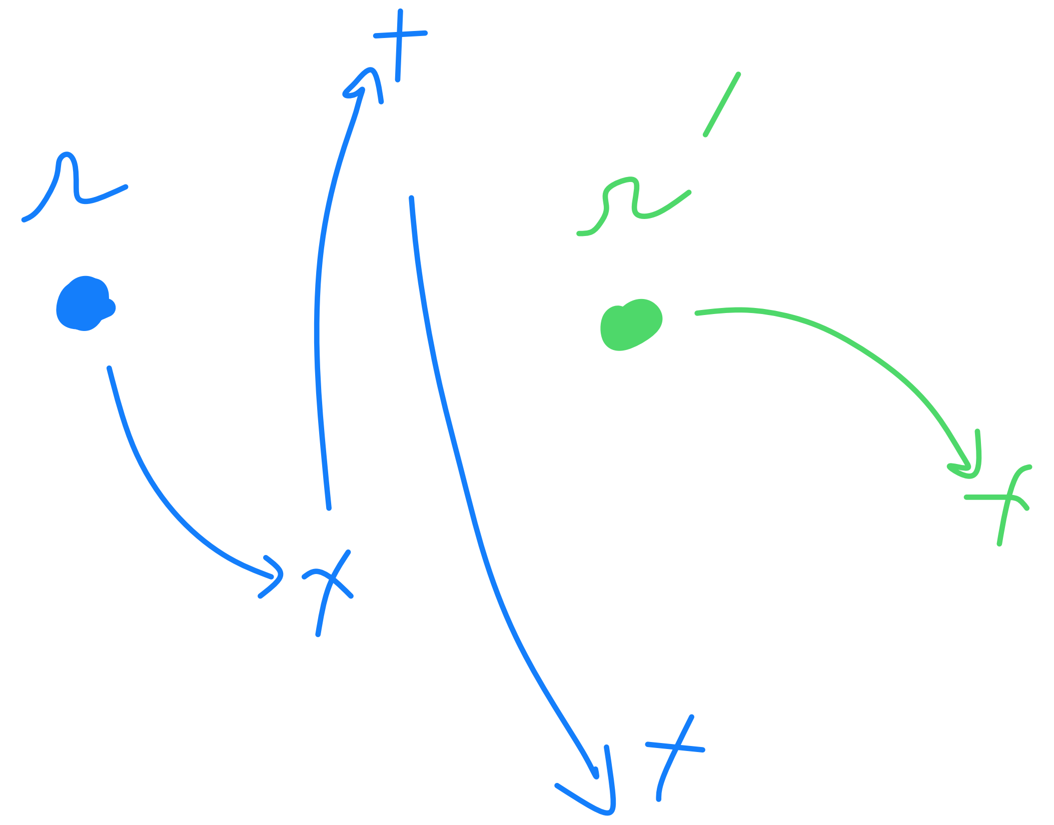
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

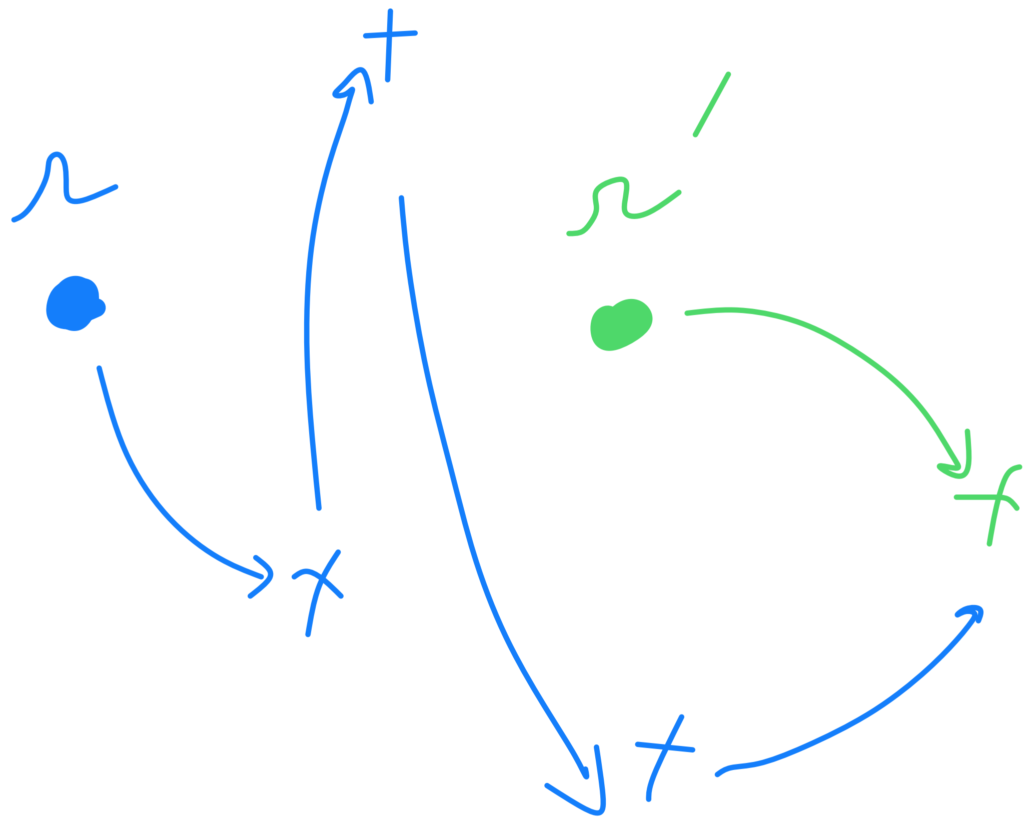
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

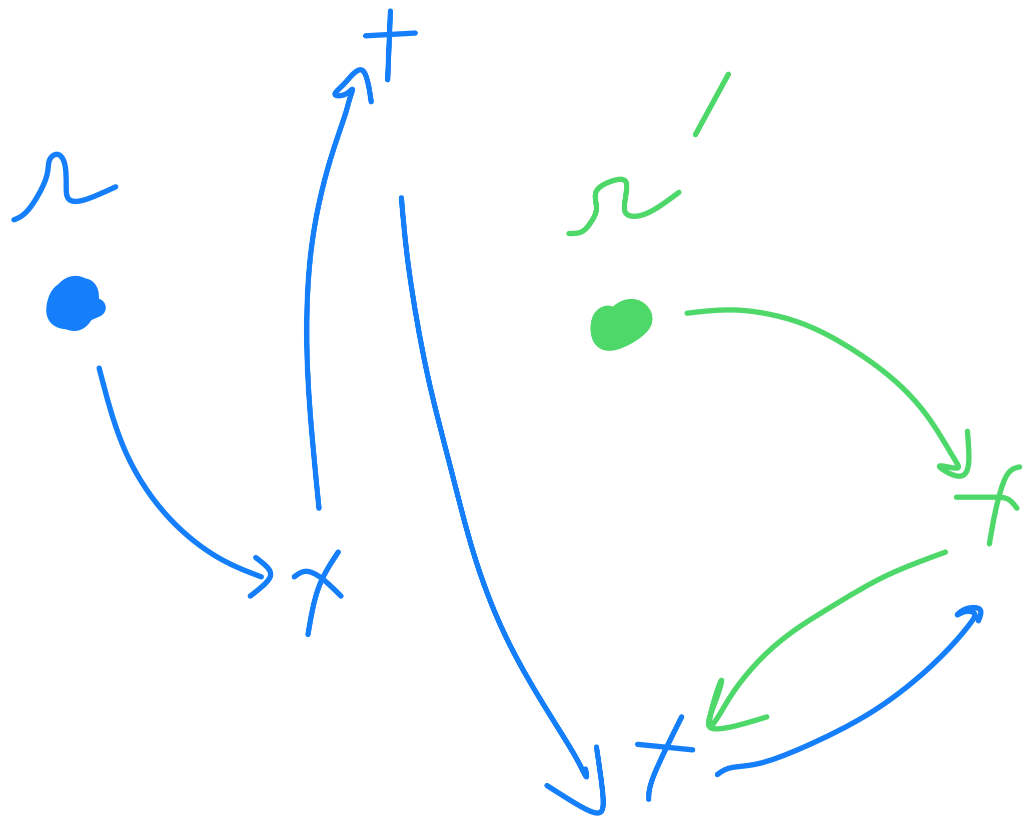
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



Impossibility result

[SSS2020]

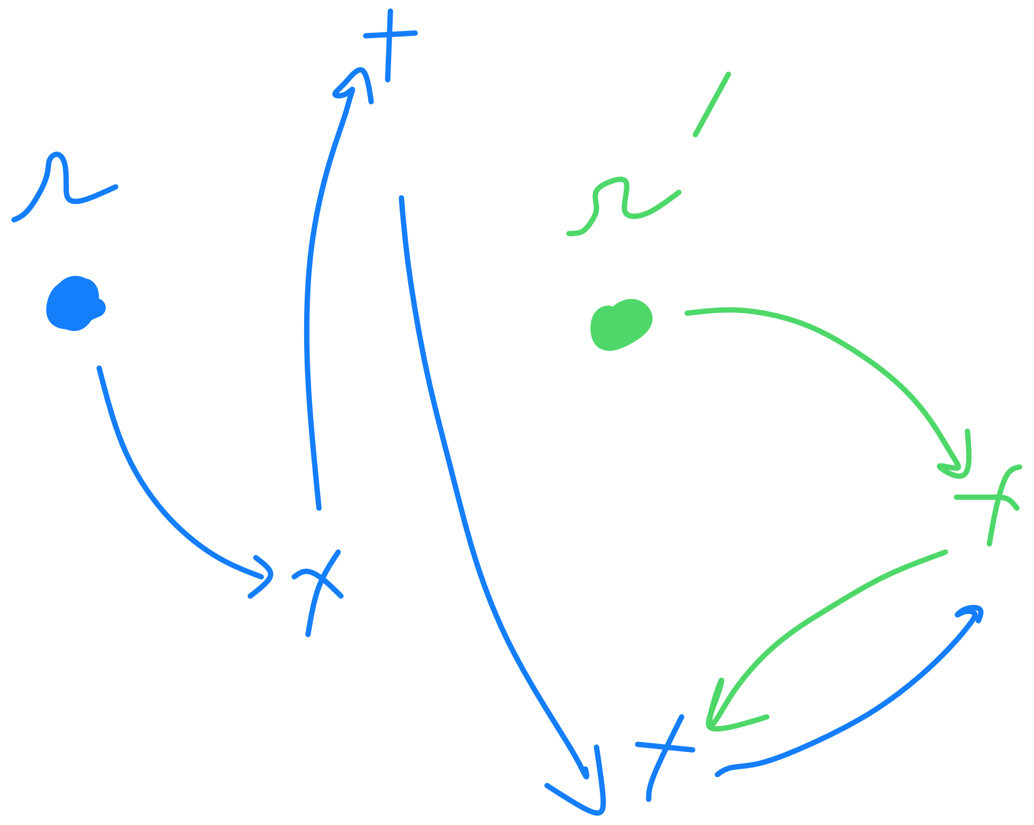
***Theorem:** SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



The scheduler is fair otherwise there exists an execution where only r (or r') is activated, without moving to the other robot, contradicting the previous Lemma.

Impossibility result

[SSS2020]

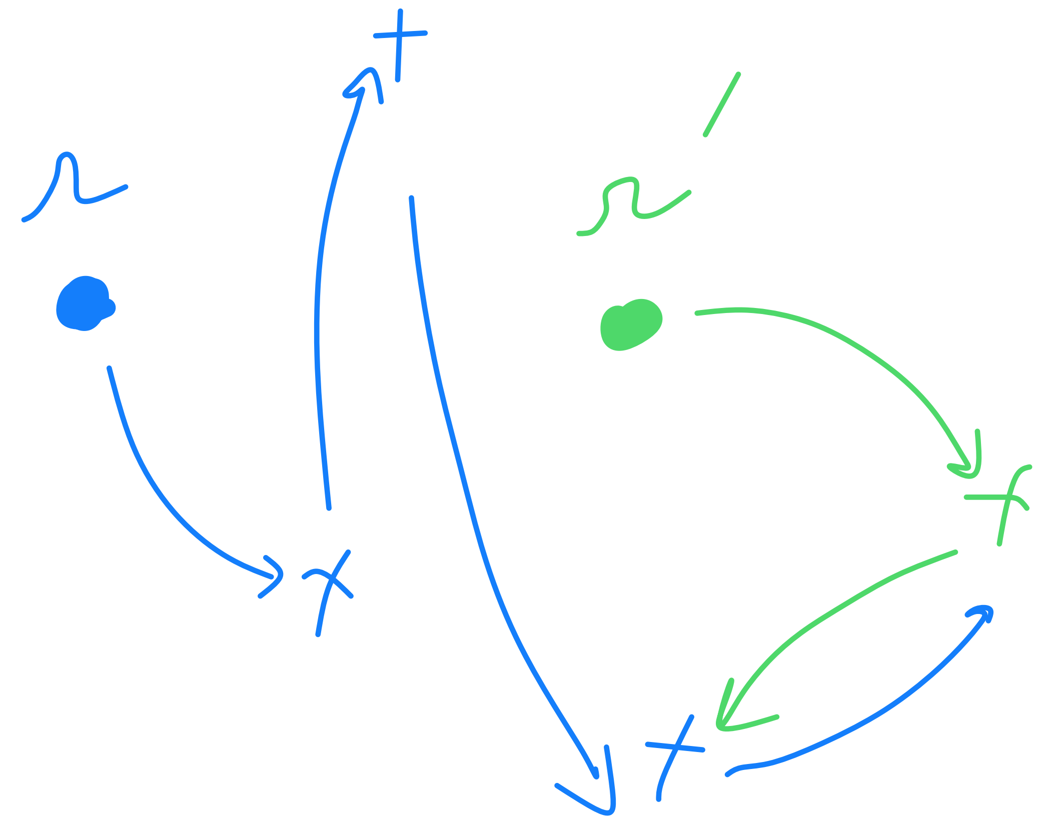
Theorem: *SUIR is not Solvable in SSYNC (even with lights and common coordinate system)*

Let A be an algorithm that solves the SUIR problem with lights (infinite memory and communication capabilities)

Let r and r' be the two robots.

Consider the following scheduler:

- If r is dictated to move to $p \neq r'$, then activate only r
- Otherwise, If r' is dictated to move to $p \neq r$, then activate only r'
- Otherwise, activate both robots.



The scheduler is fair otherwise there exists an execution where only r (or r') is activated, without moving to the other robot, contradicting the previous Lemma.

After each round the robots are not gathered.

Solution with 2 robots

[SSS2020]

Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

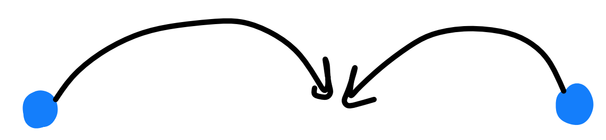
If they decide to move to the middle

Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

If they decide to move to the middle

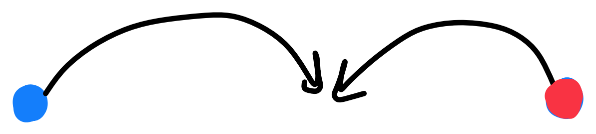


Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

If they decide to move to the middle



Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

If they decide to move to the middle

Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

If they decide to move to the middle

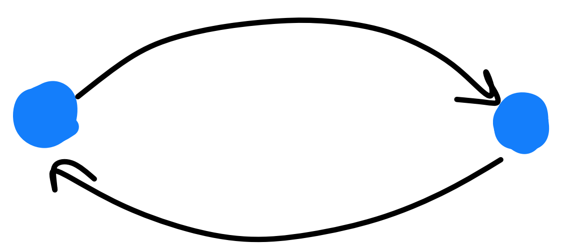
Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

If they decide to move to the middle

If they decide to move to the other



Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

If they decide to move to the middle

If they decide to move to the other

Solution with 2 robots

[SSS2020]

Maybe it seems impossible at first because both robots see the same thing, hence do the same thing:

If they decide to move to the middle

If they decide to move to the other

Solution with 2 robots

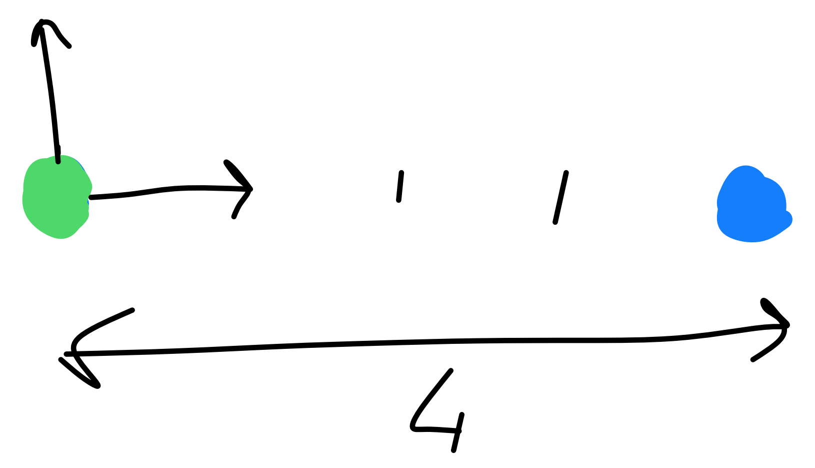
[SSS2020]

But if r moves to the middle "alone", r’s view has changed!!

Solution with 2 robots

[SSS2020]

But if r moves to the middle "alone", r's view has changed!!

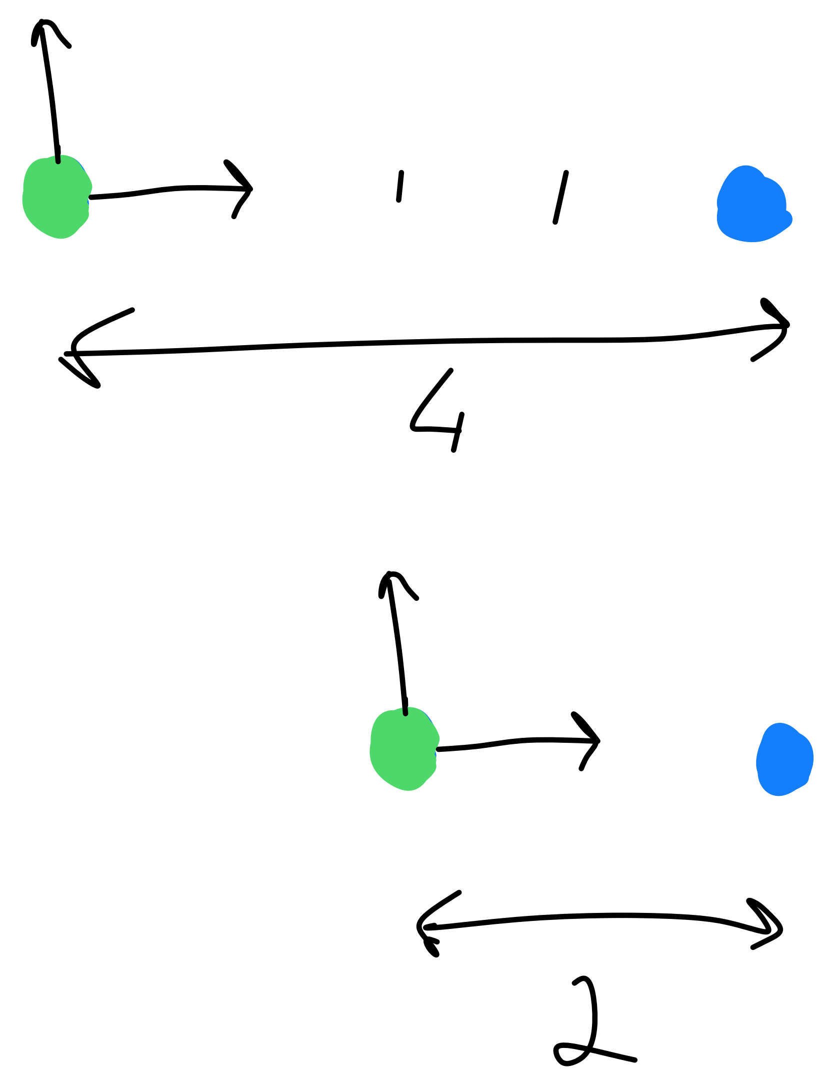


Solution with 2 robots

[SSS2020]

But if r moves to the middle "alone", r's view has changed!!

After



Solution with 2 robots

[SSS2020]

But if r moves to the middle "alone", r's view has changed!!

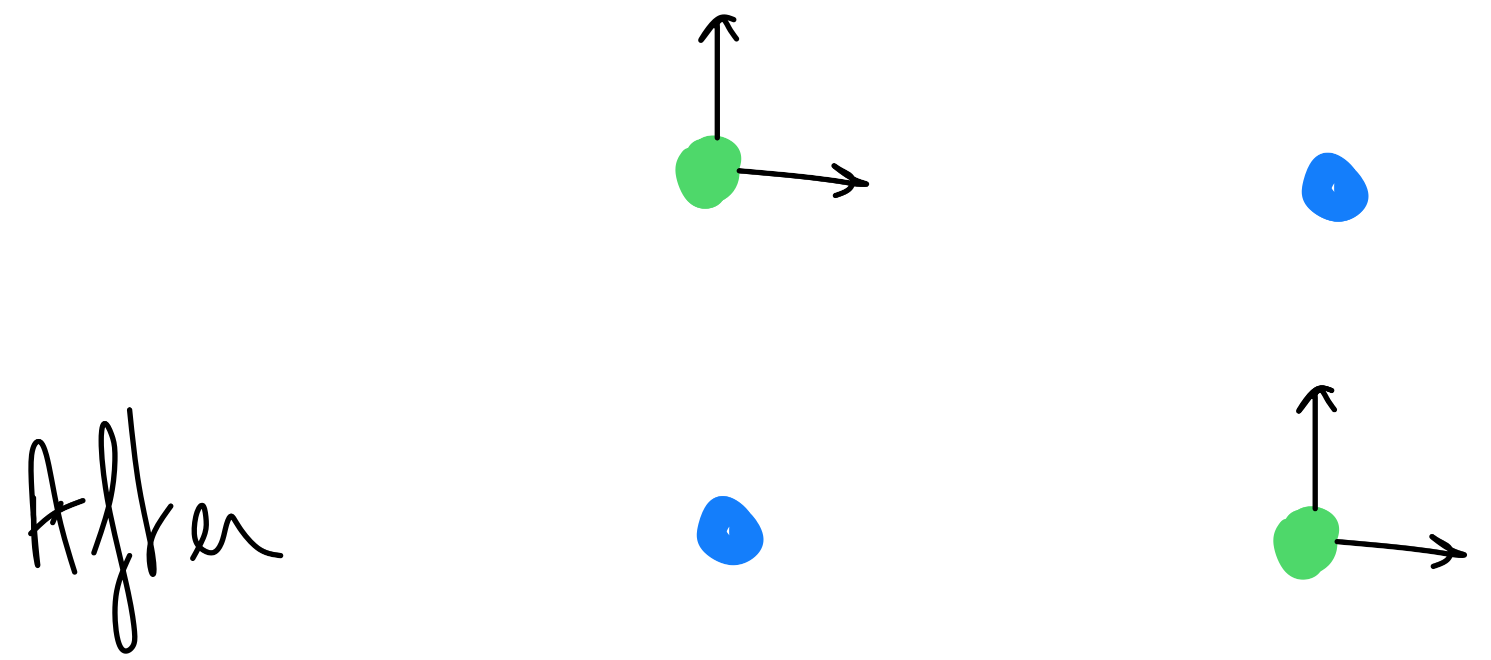
And if robots swap their positions, their views have changed!!

Solution with 2 robots

[SSS2020]

But if r moves to the middle "alone", r's view has changed!!

And if robots swap their positions, their views have changed!!



Solution with 2 robots

[SSS2020]

But if r moves to the middle "alone", r’s view has changed!!

And if robots swap their positions, their views have changed!!

Solution with 2 robots

[SSS2020]

But if r moves to the middle "alone", r’s view has changed!!

And if robots swap their positions, their views have changed!!

So the key is to use the distance between the robots and the orientation of the axe, because those are fixed.

Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

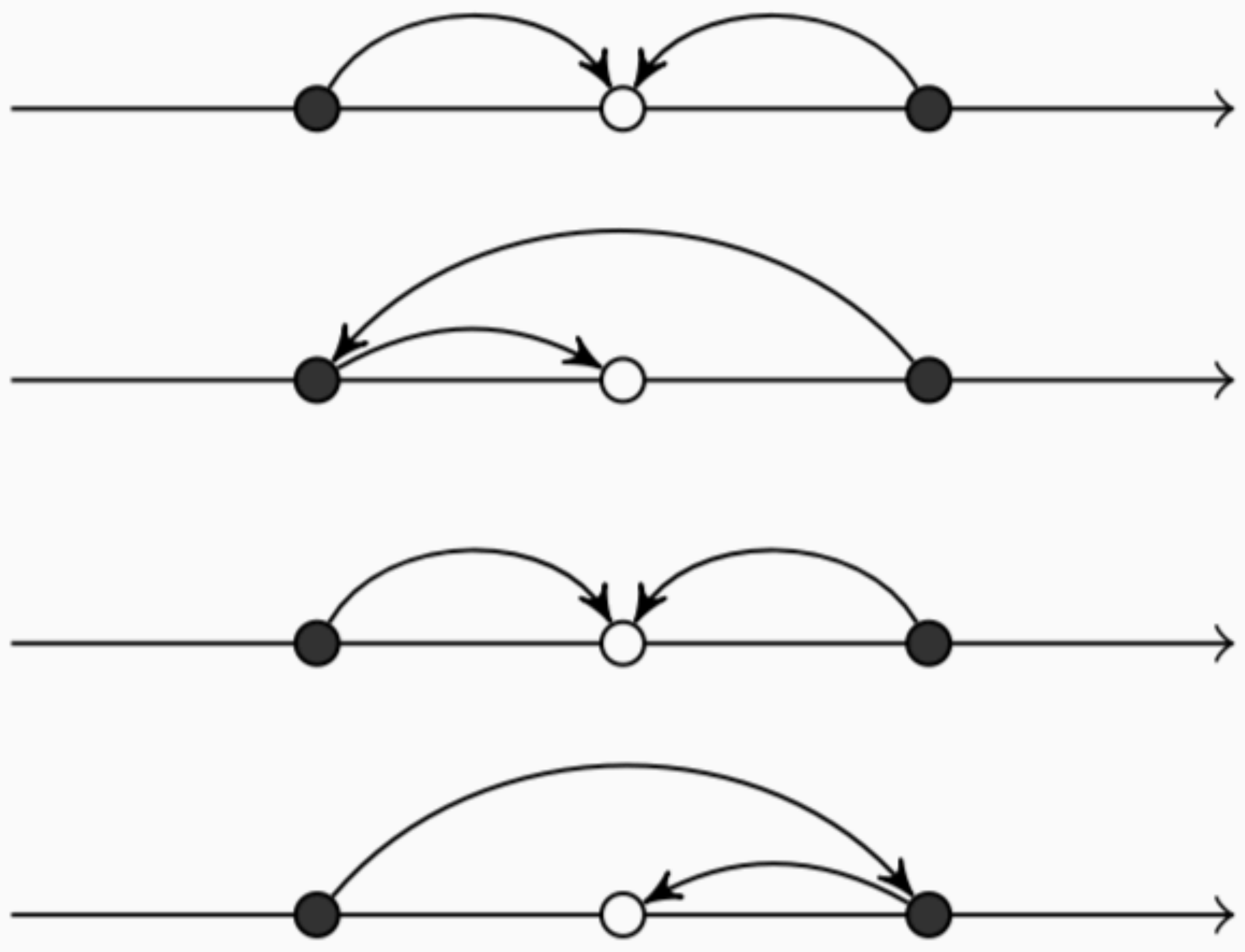
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$



Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

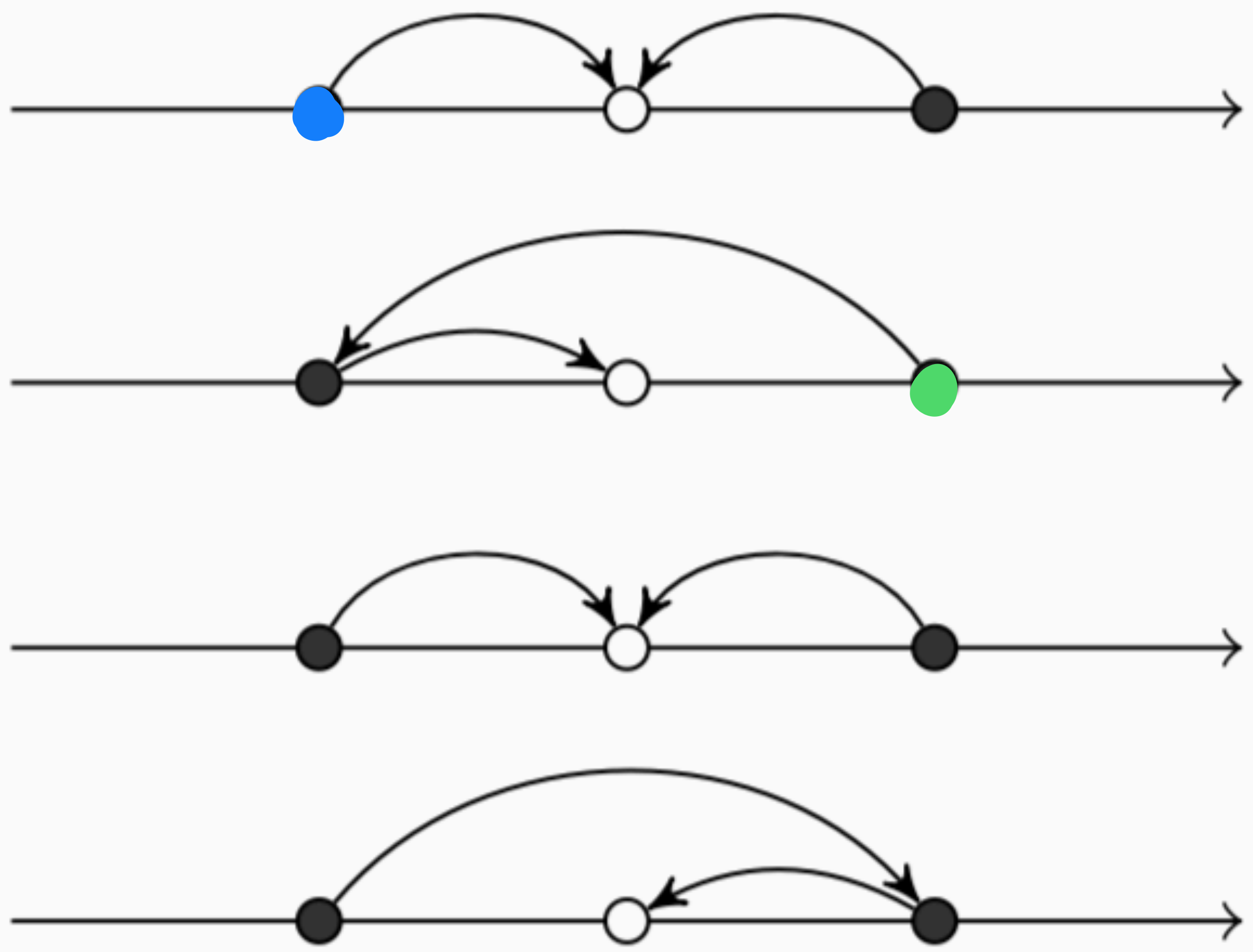
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

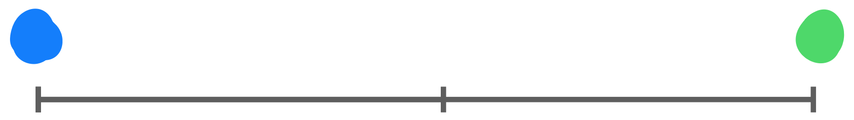
case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$



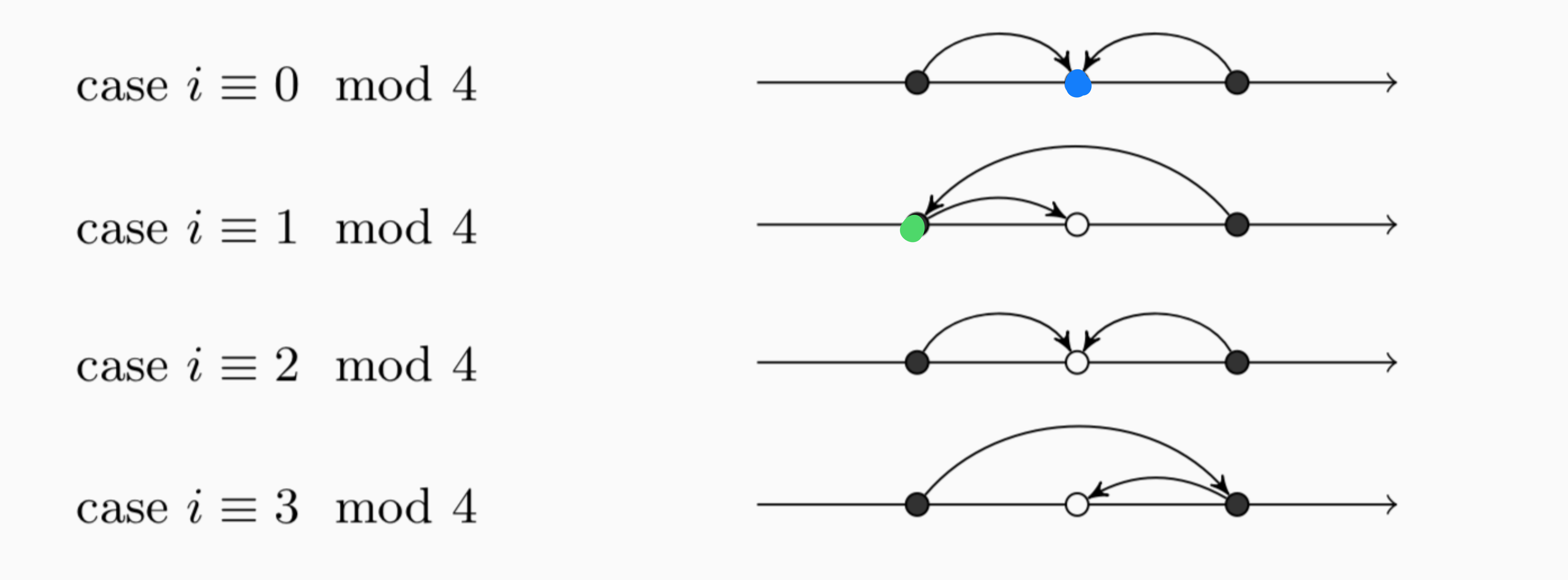


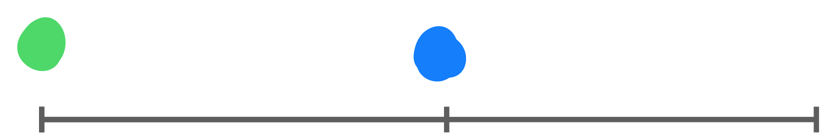
Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$





Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

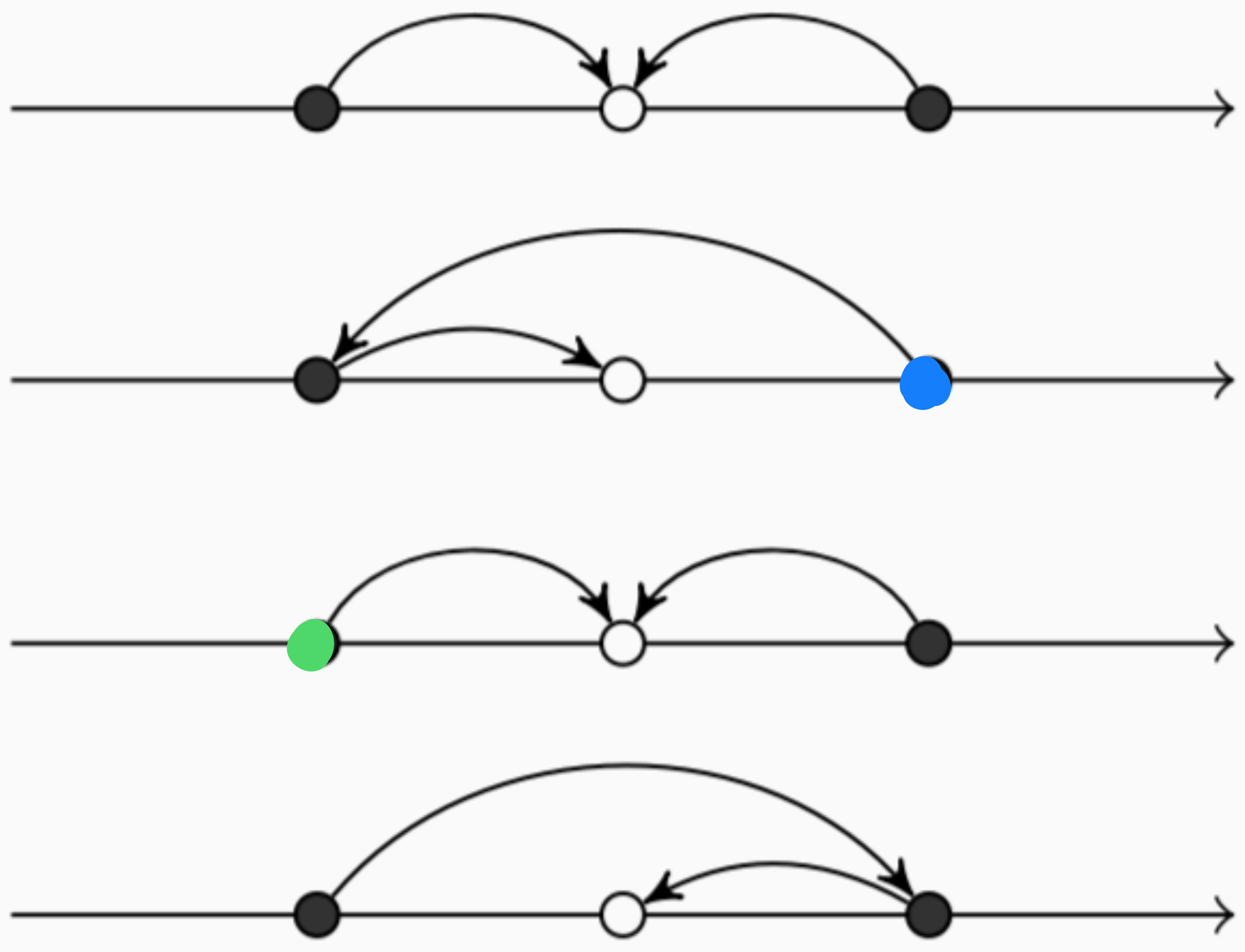
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

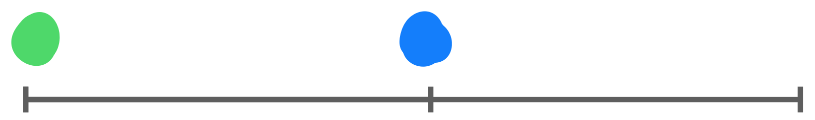
case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$





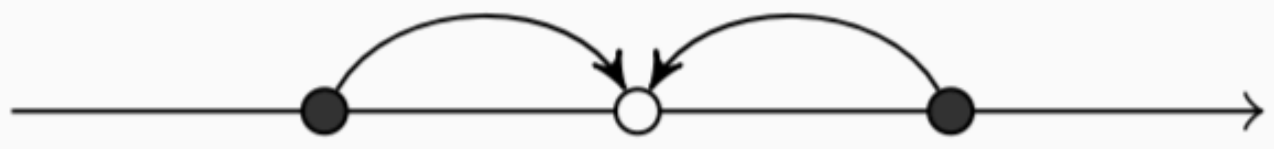
Solution with 2 robots

[SSS2020]

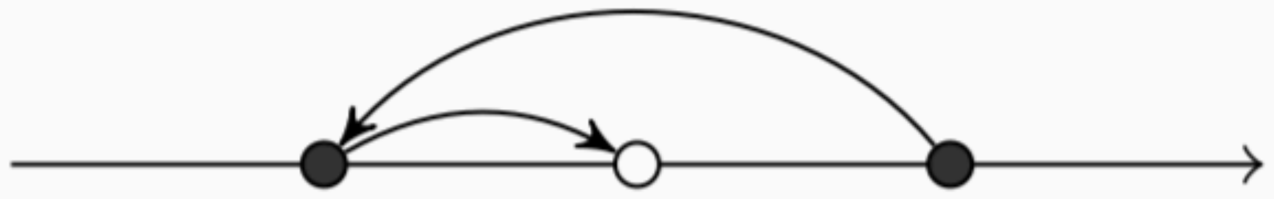
If d is the distance between the two robots (seen by robot r, executing the algorithm)

Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

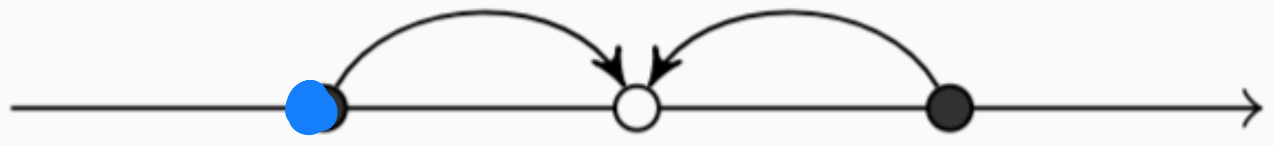
case $i \equiv 0 \mod 4$



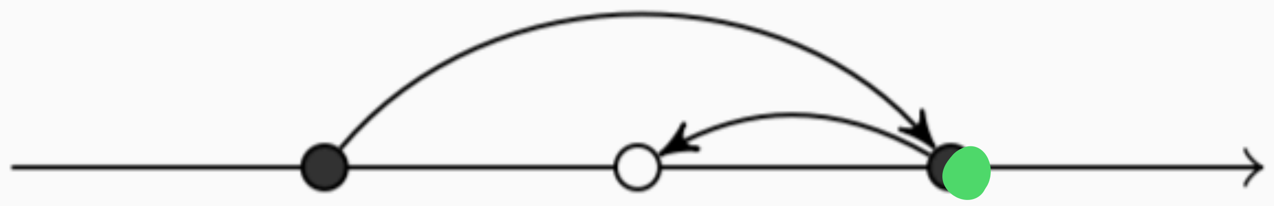
case $i \equiv 1 \mod 4$

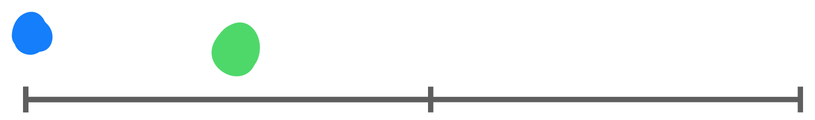


case $i \equiv 2 \mod 4$



case $i \equiv 3 \mod 4$





Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

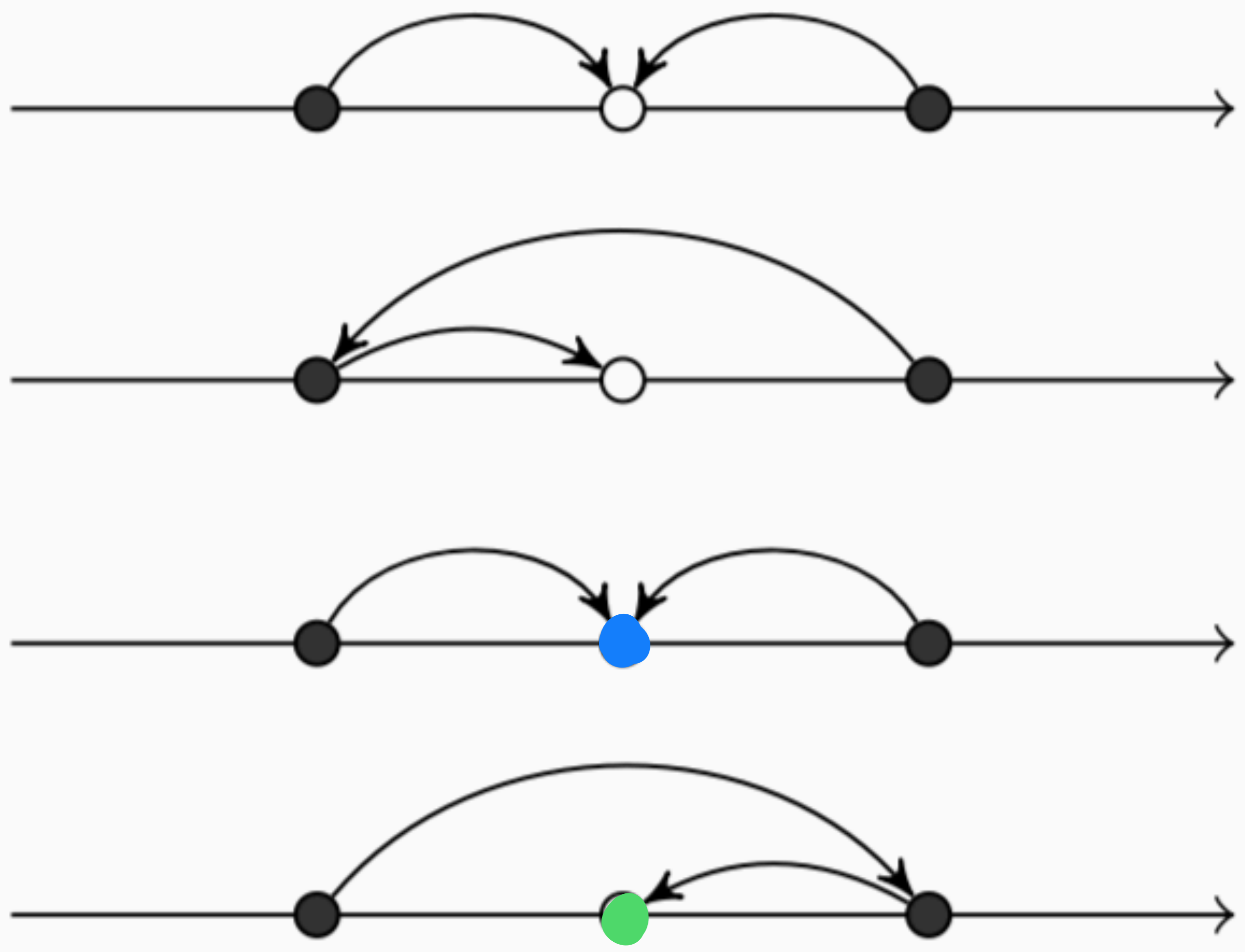
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

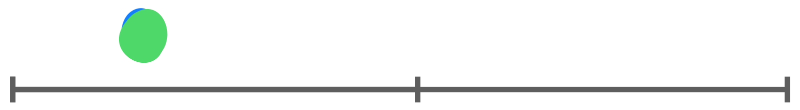
case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$





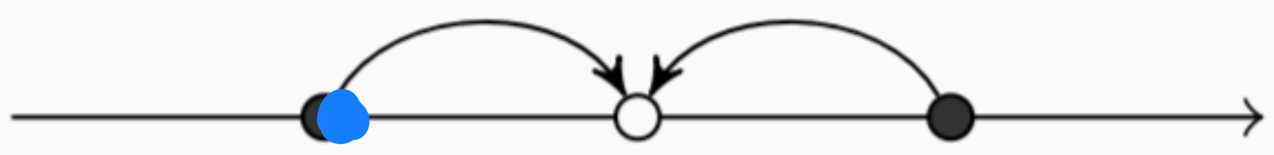
Solution with 2 robots

[SSS2020]

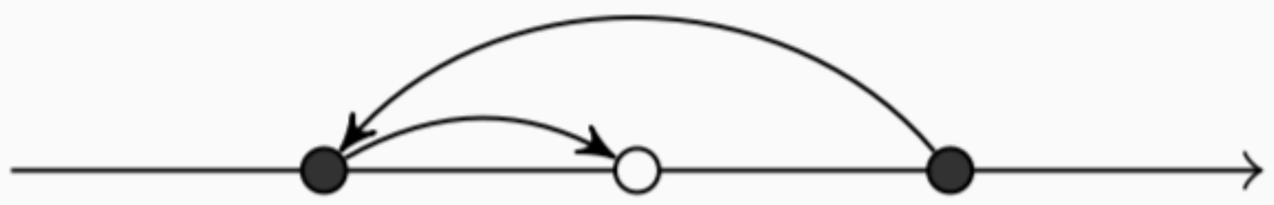
If d is the distance between the two robots (seen by robot r, executing the algorithm)

Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

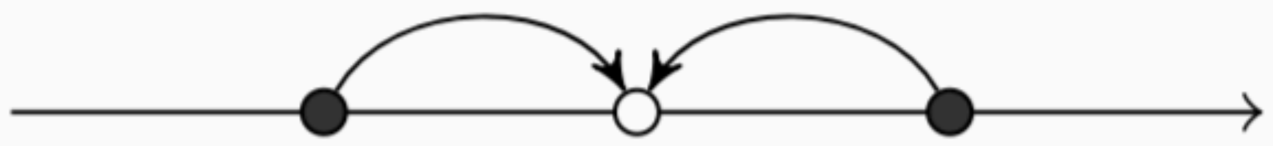
case $i \equiv 0 \mod 4$



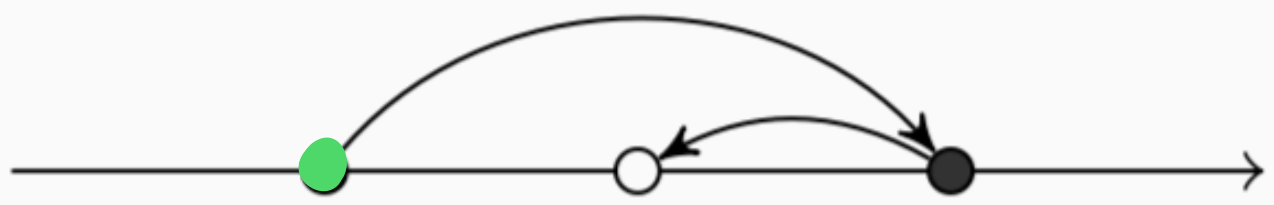
case $i \equiv 1 \mod 4$



case $i \equiv 2 \mod 4$



case $i \equiv 3 \mod 4$



Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

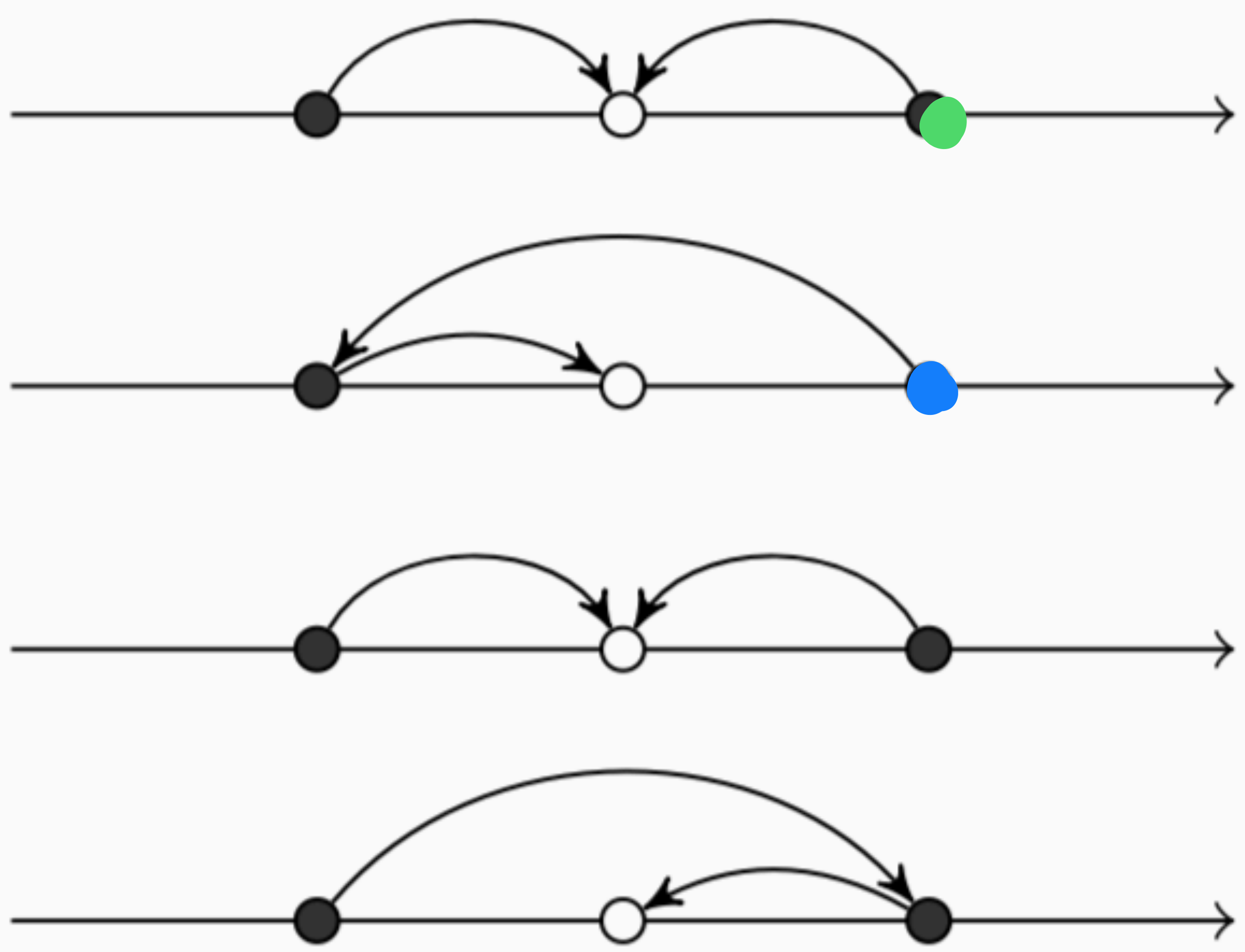
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$



Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

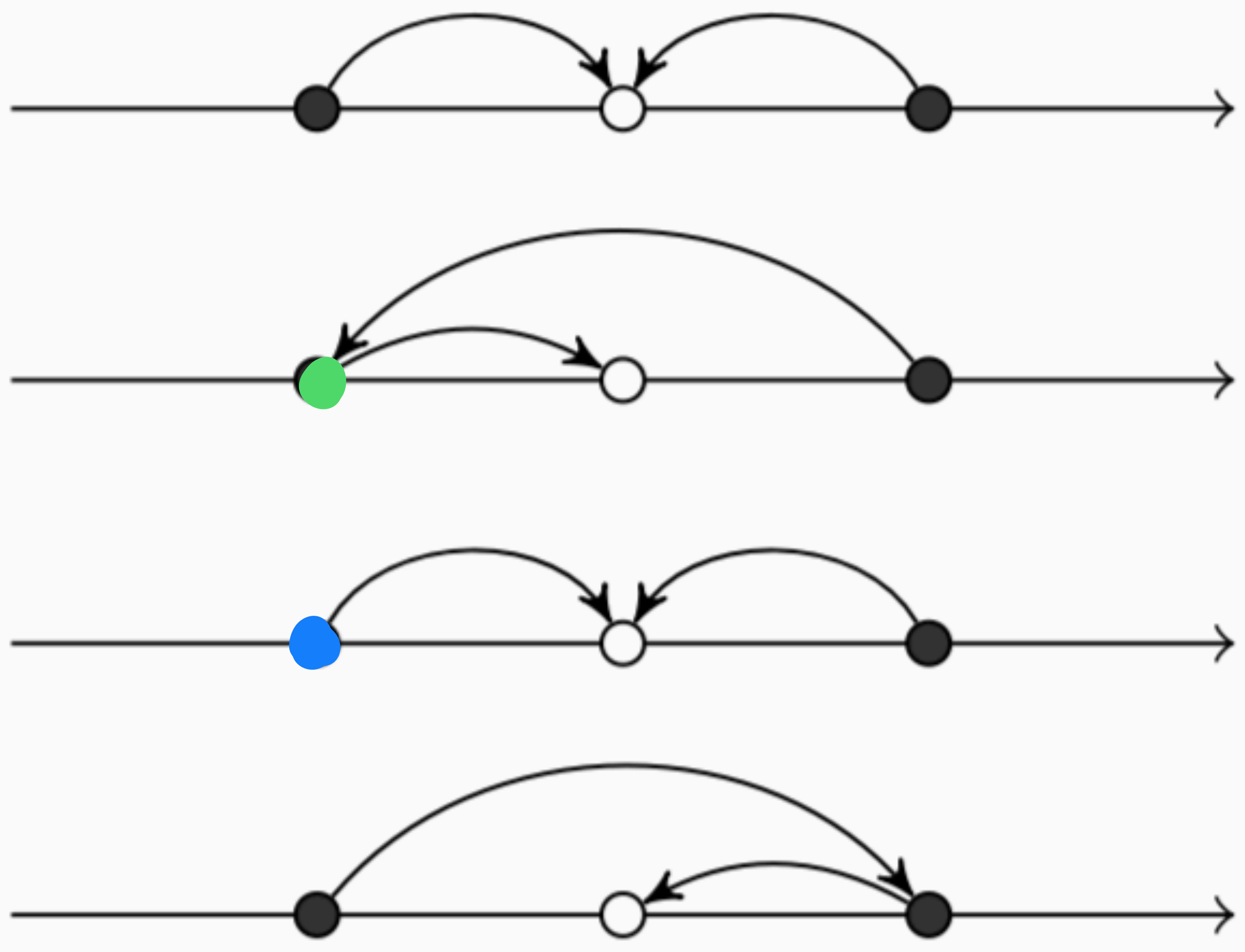
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$



Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

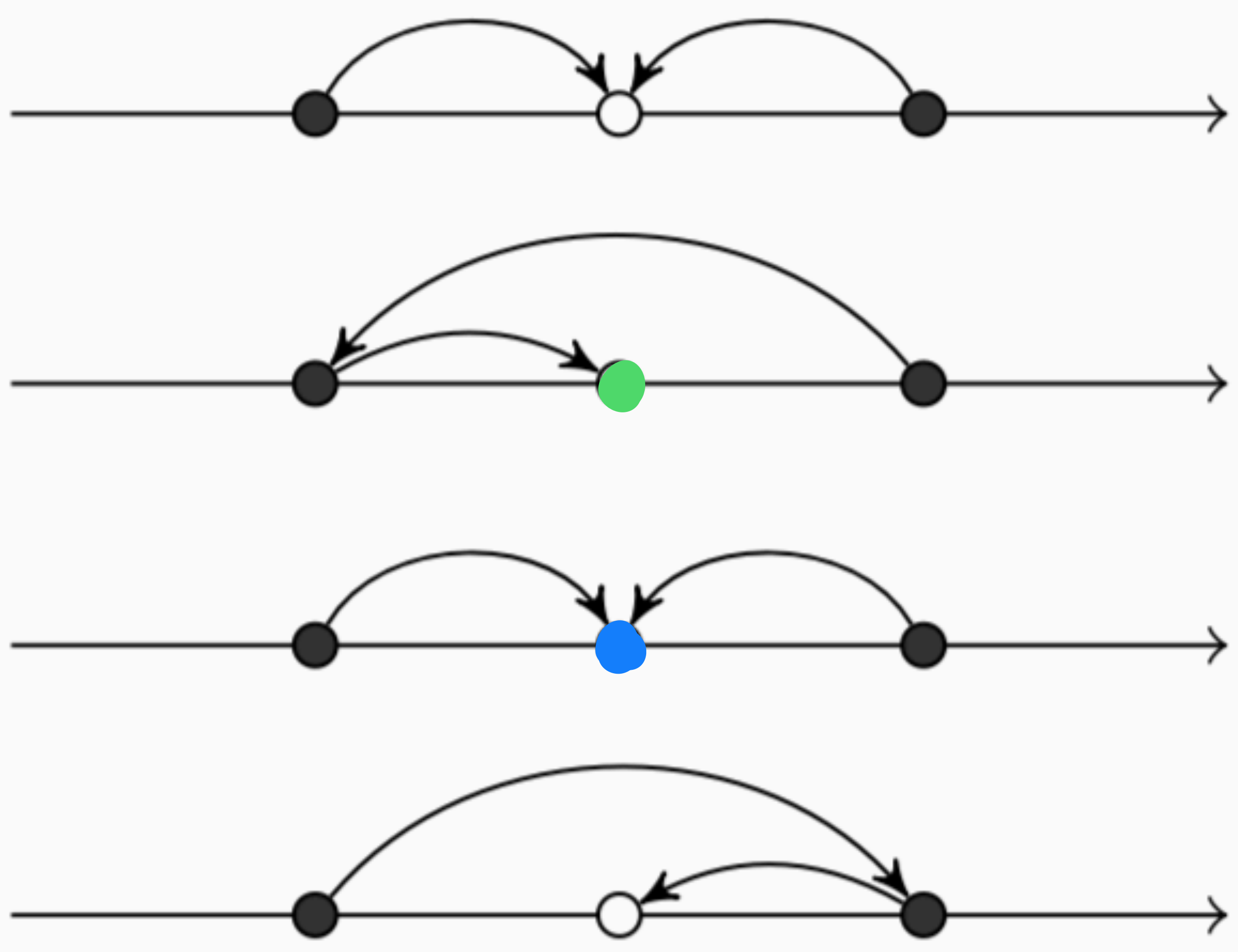
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$

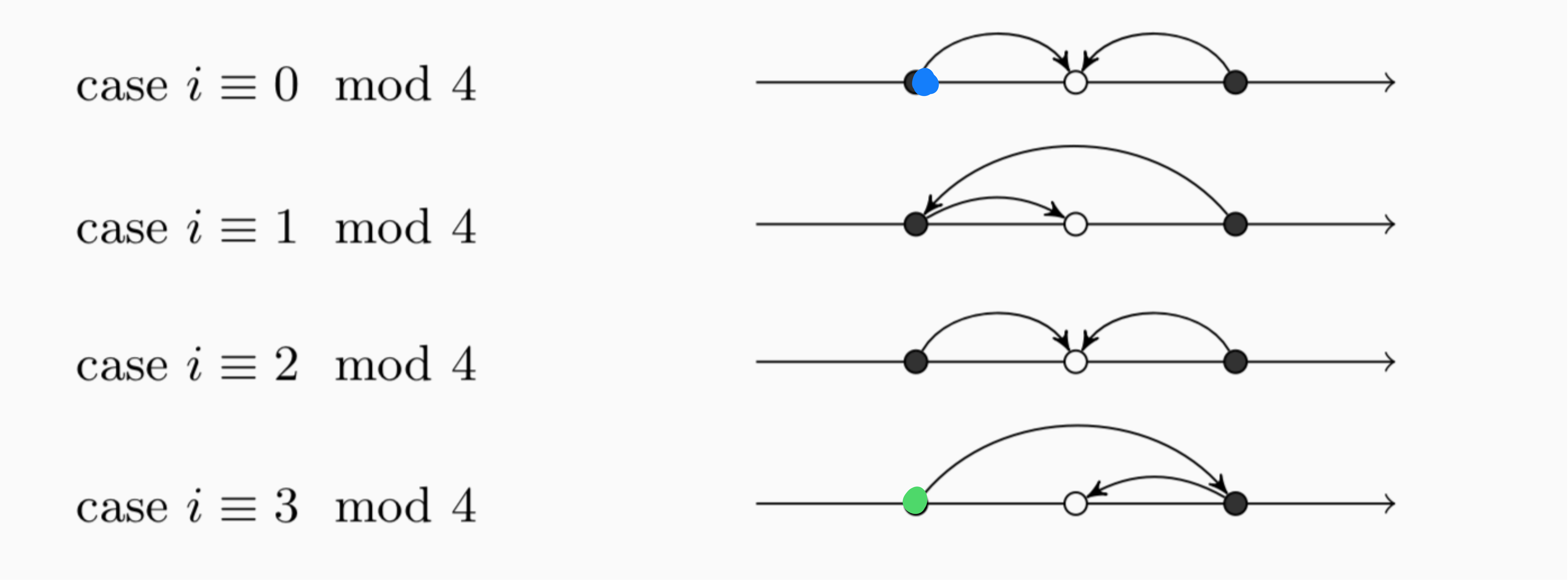


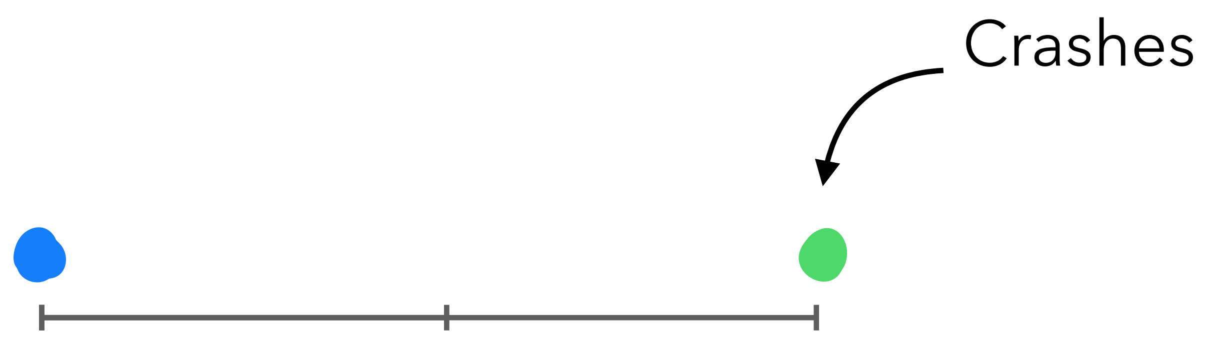
Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$





Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

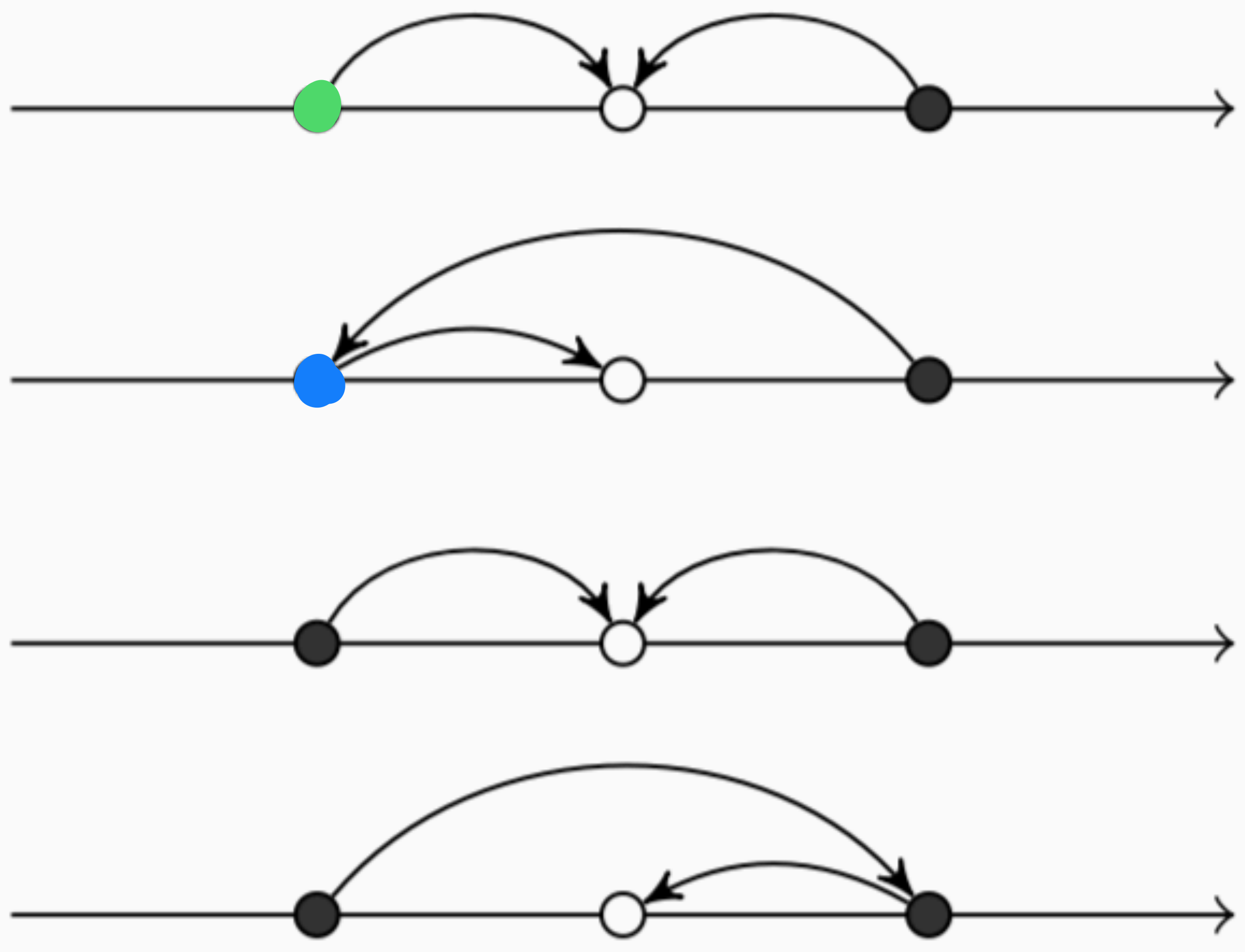
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$



Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

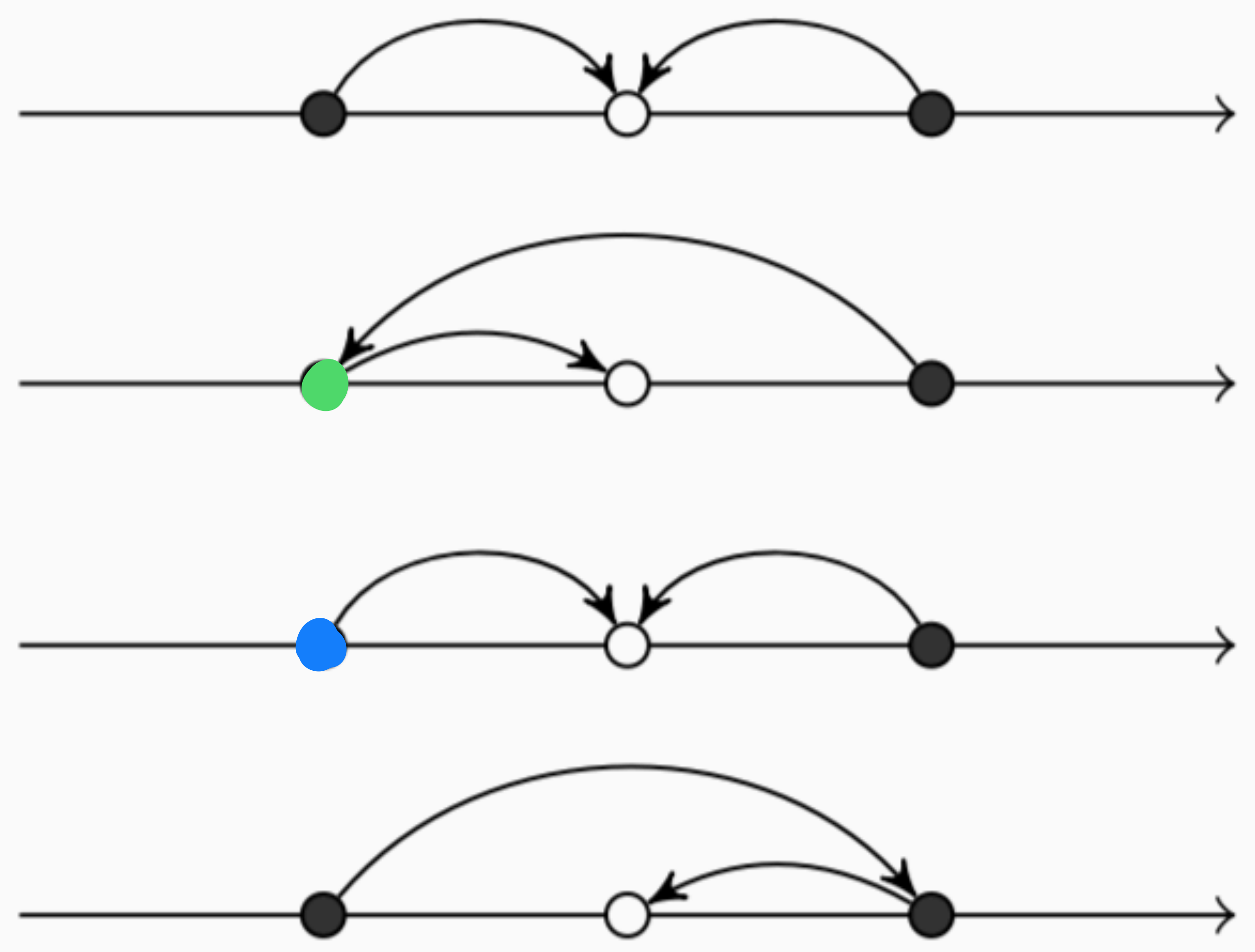
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

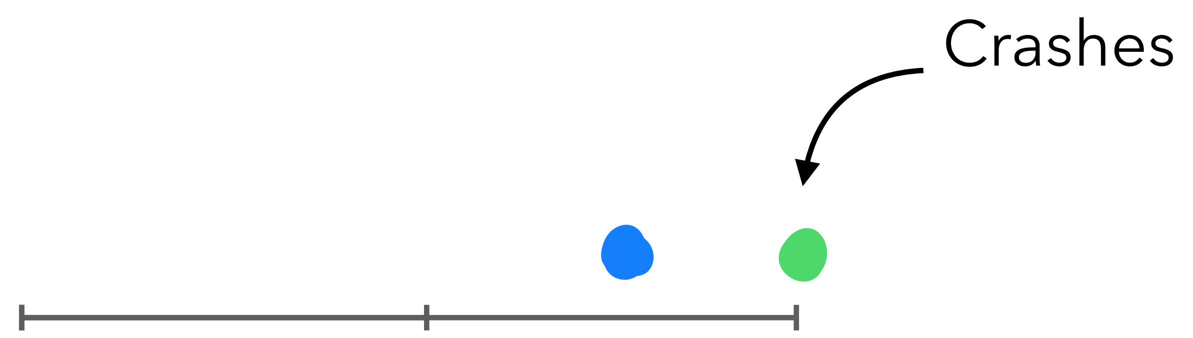
case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$





Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

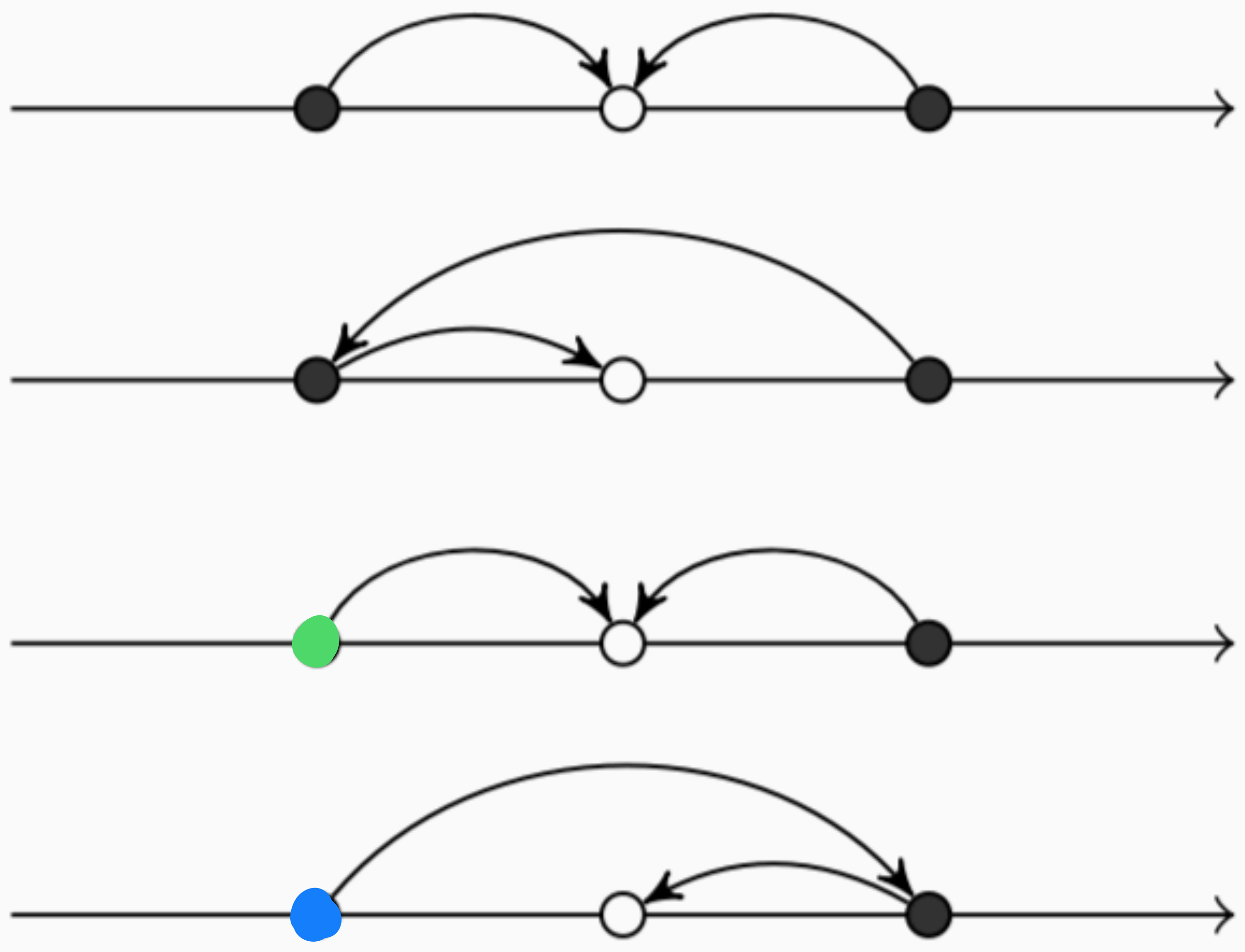
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

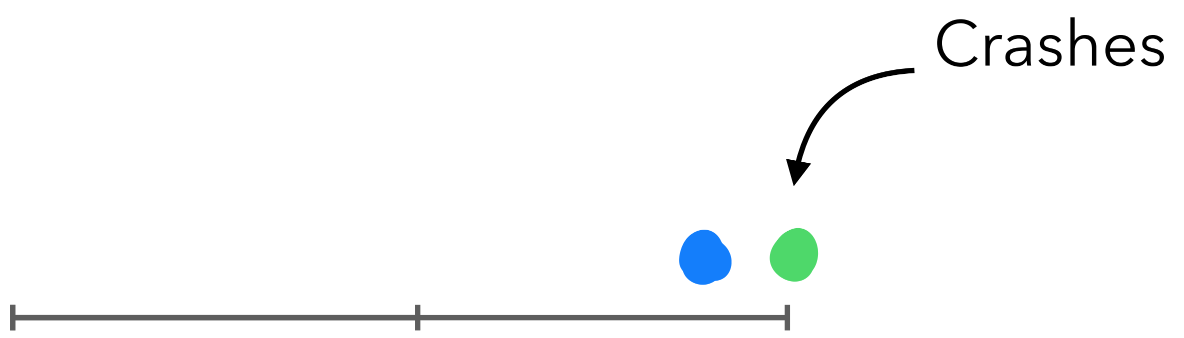
case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

case $i \equiv 3 \mod 4$





Solution with 2 robots

[SSS2020]

If d is the distance between the two robots (seen by robot r, executing the algorithm)

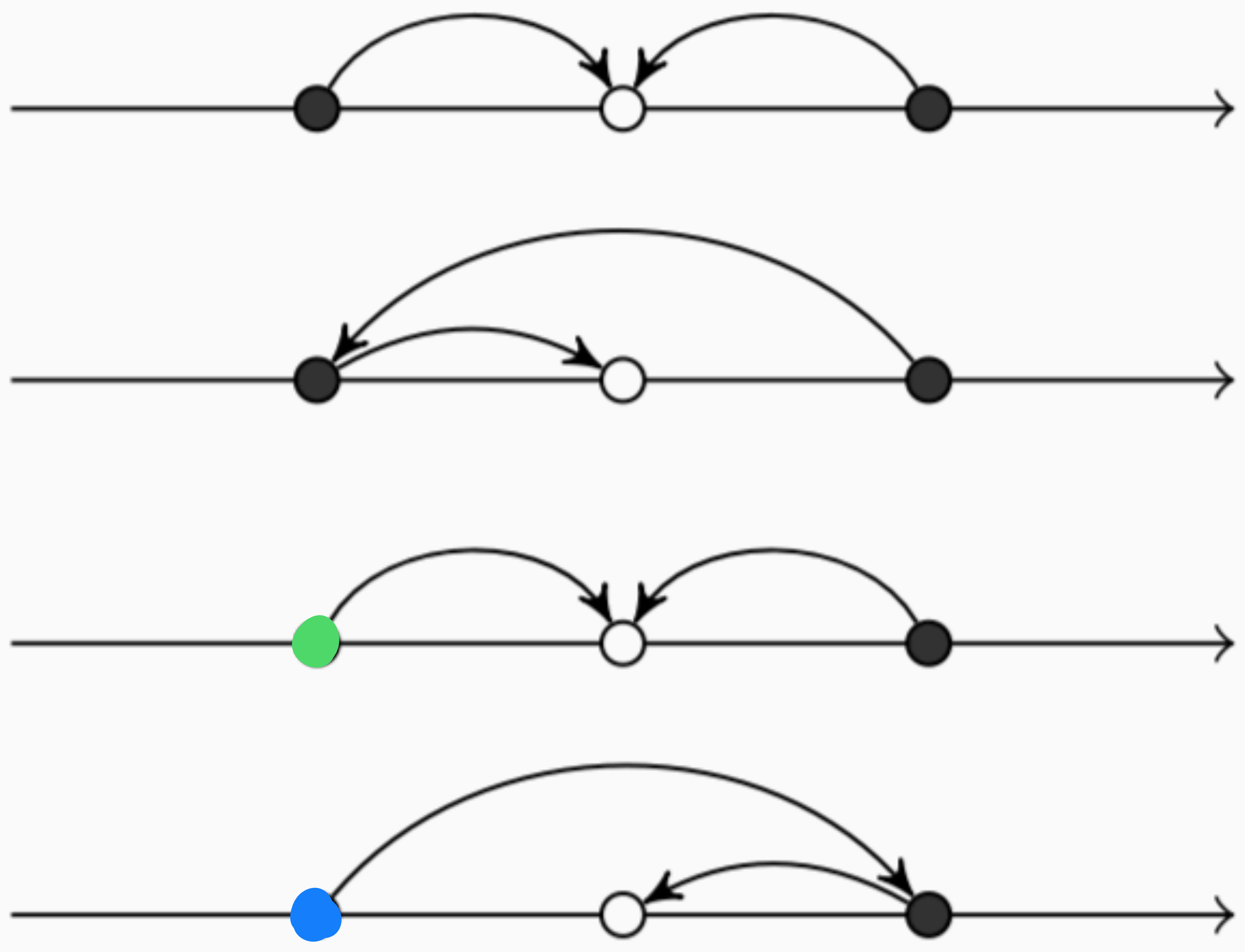
Let $i \in \mathbb{Z}$ such that $d \in [2^{-i}, 2^{1-i})$

case $i \equiv 0 \mod 4$

case $i \equiv 1 \mod 4$

case $i \equiv 2 \mod 4$

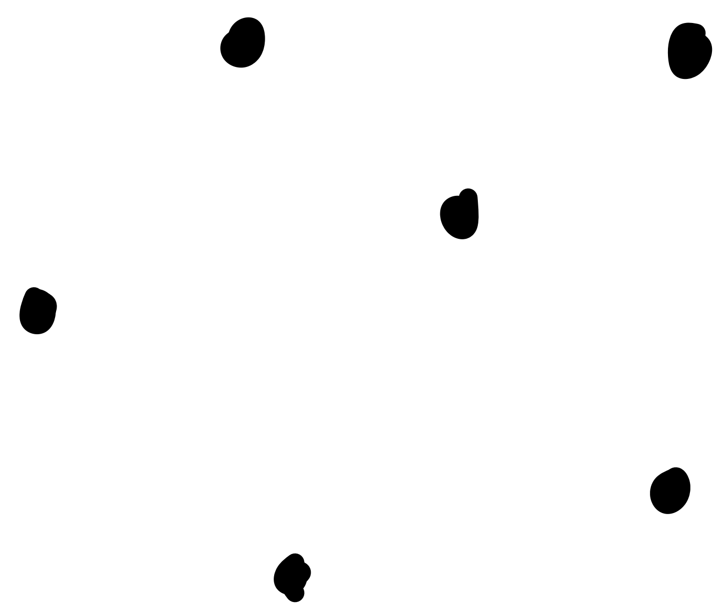
case $i \equiv 3 \mod 4$



Solution with $n > 2$ robots

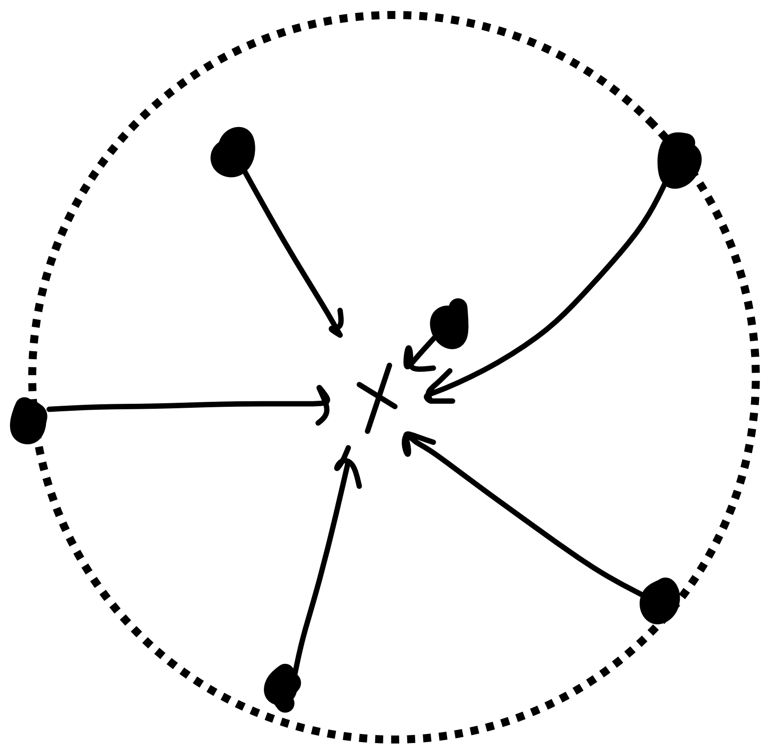
How to start

n > 2



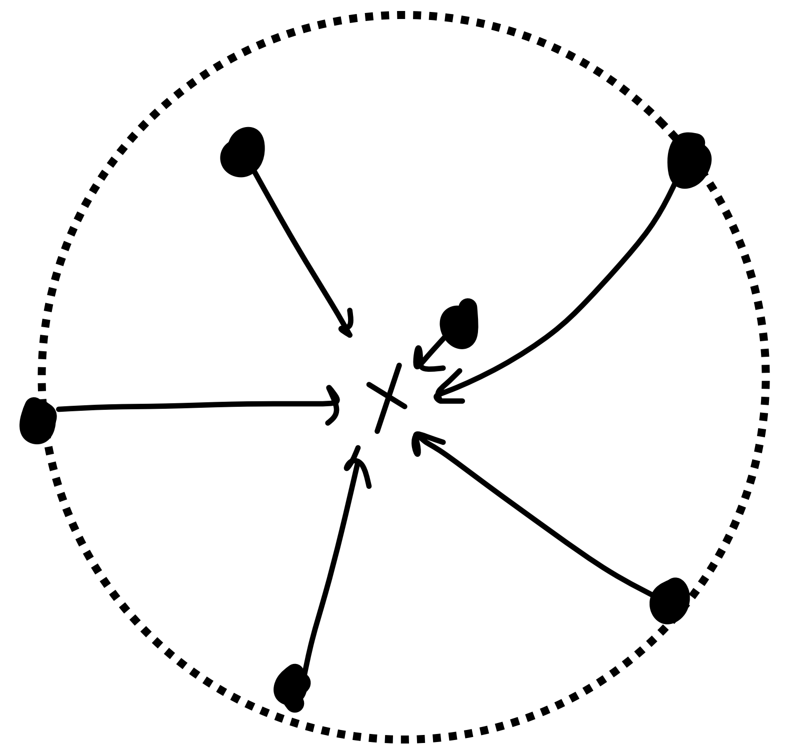
How to start

n > 2



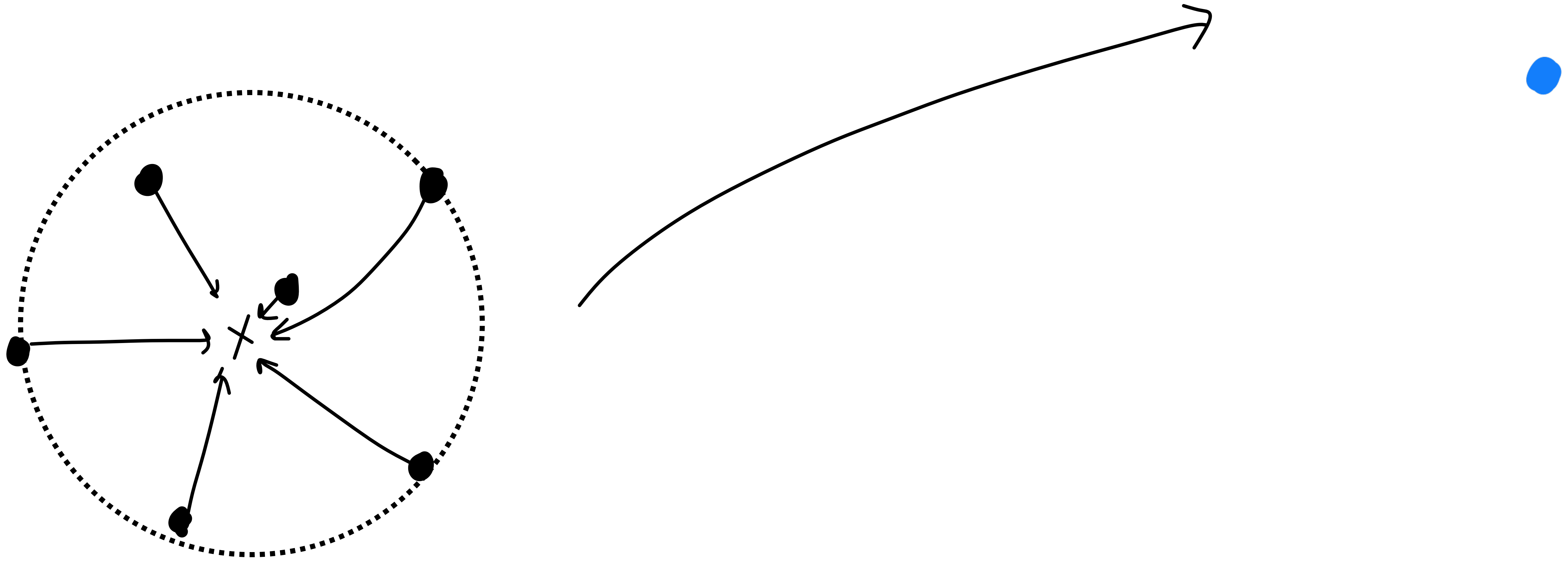
How to start

n > 2



How to start

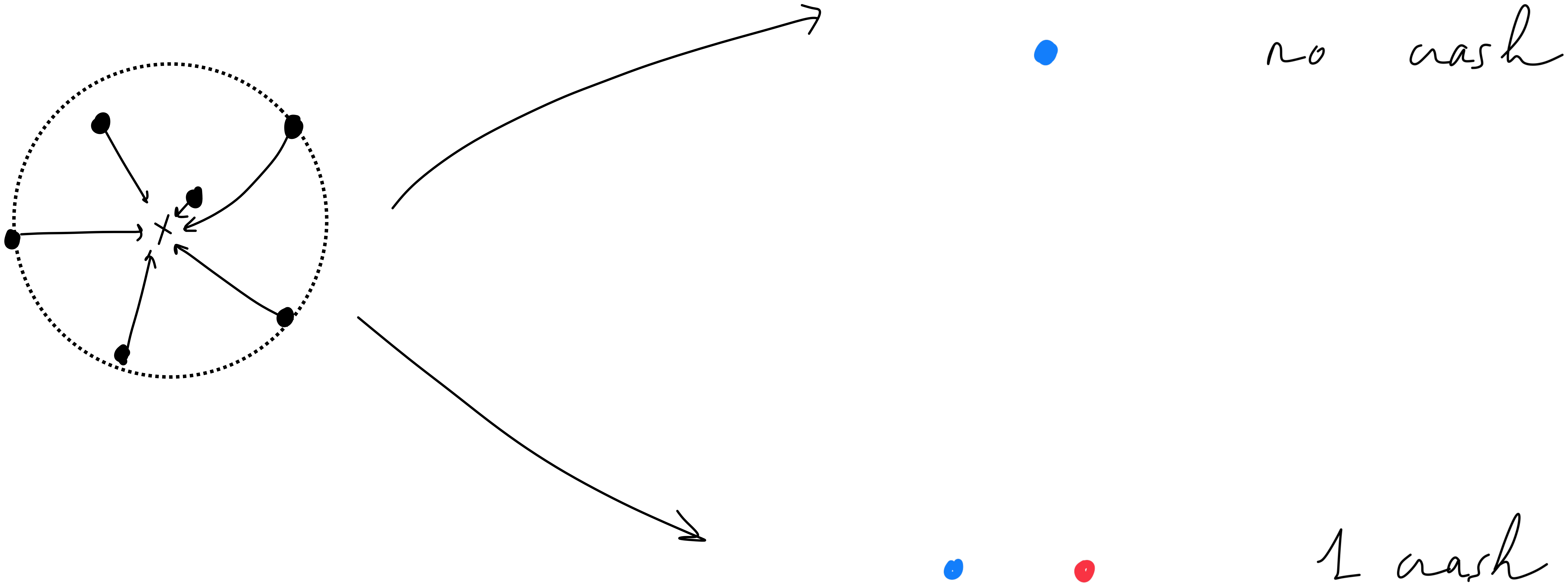
n > 2



no cash

How to start

n > 2



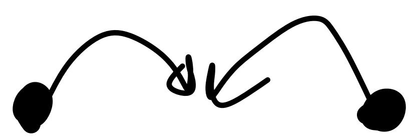
What we would like to do

- If all the robots are correct we want all of them to execute move to middle

What we would like to do

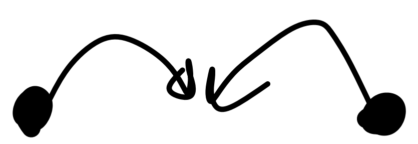
- If all the robots are correct we want all of them to execute move to middle

i = 1



What we would like to do

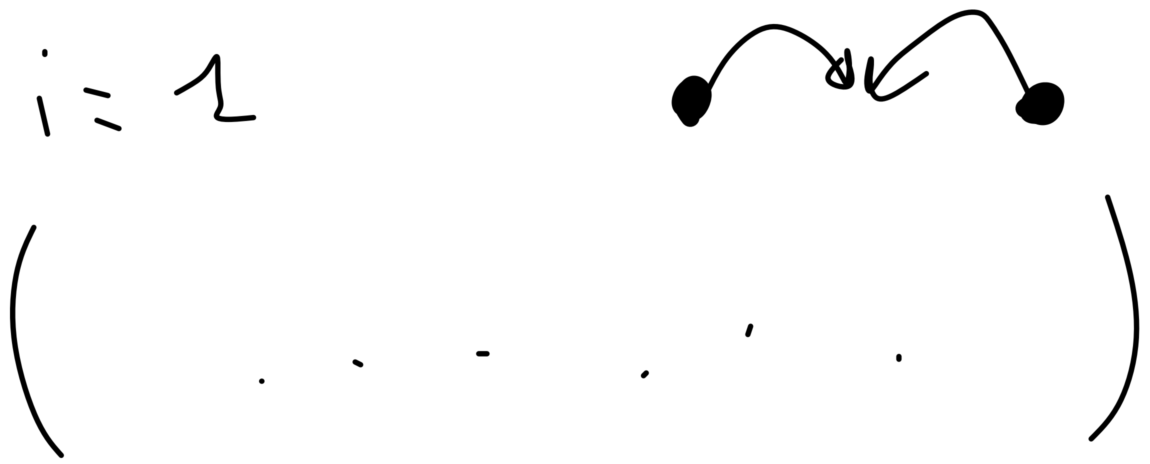
- If all the robots are correct we want all of them to execute move to middle

$i = 1$  ← works if all robots have the same level

What we would like to do

- If all the robots are correct we want all of them to execute move to middle

$i = 1$

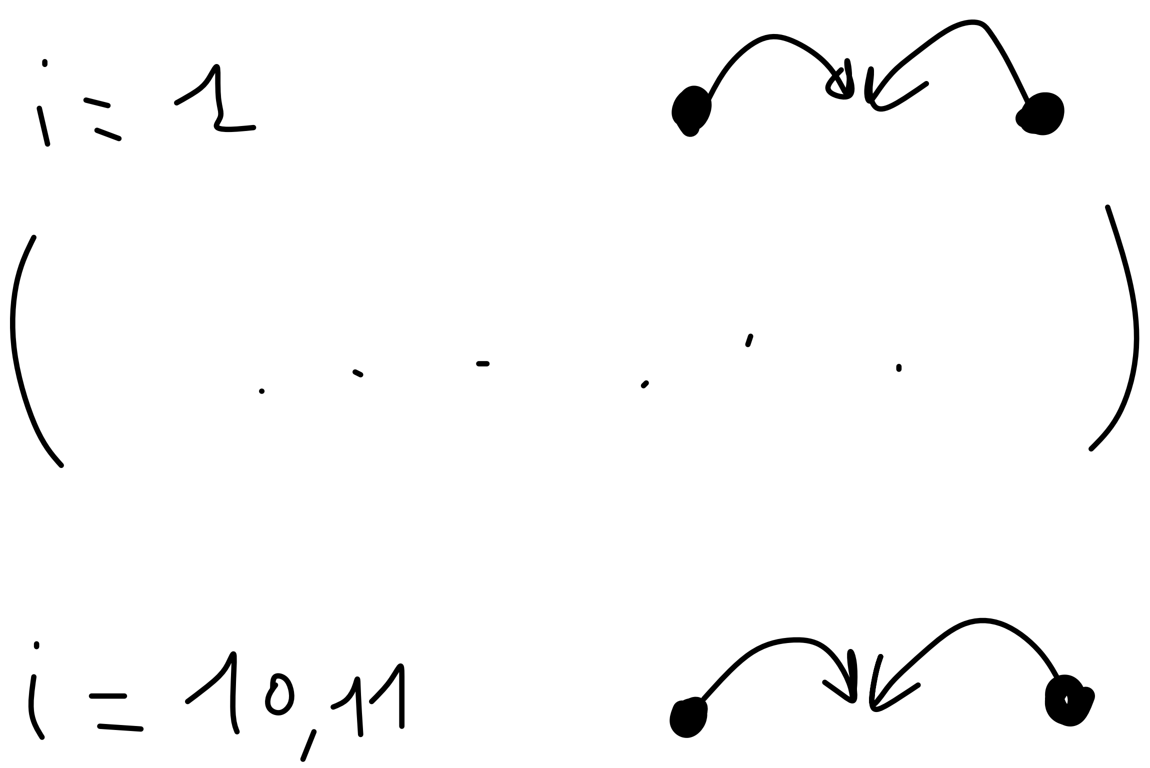


← works if all robots have the same level

What we would like to do

- If all the robots are correct we want all of them to execute move to middle

$i = 1$ ← works if all robots have the same level

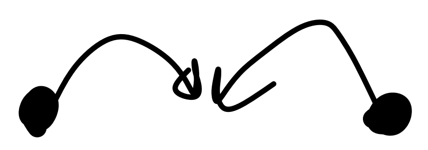


$i = 10, 11$

What we would like to do

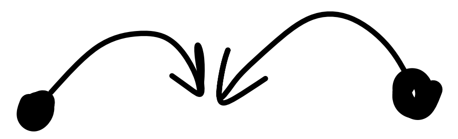
- If all the robots are correct we want all of them to execute move to middle

$i = 1$



← works if all robots have the same level

$i = 10, 11$

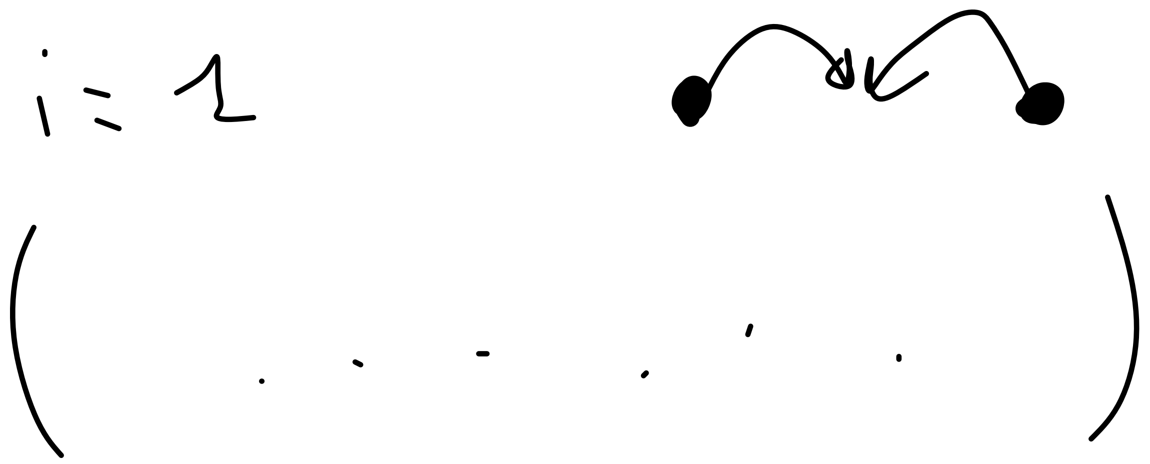


← works if Δlevel ≤ 2

What we would like to do

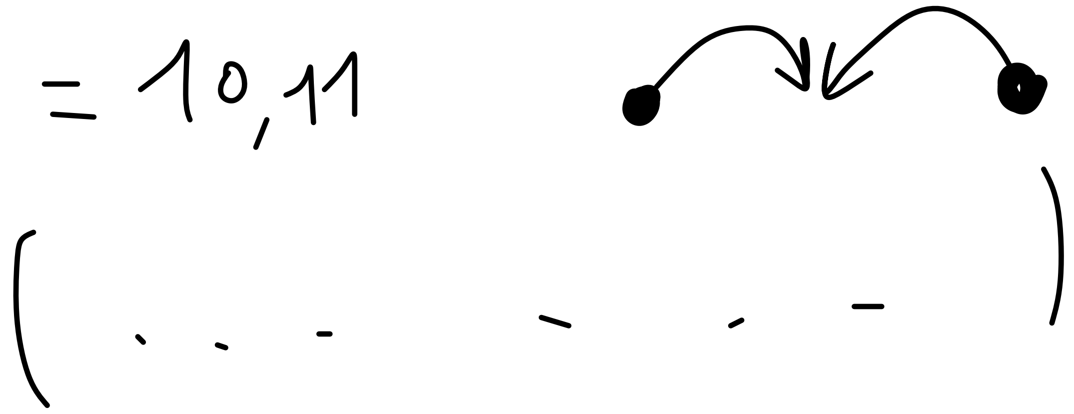
- If all the robots are correct we want all of them to execute move to middle

$i = 1$



← works if all robots have the same level

$i = 10, 11$



← works if $\Delta_{level} \leq 2$

What we would like to do

- If all the robots are correct we want all of them to execute move to middle

$i = 1$

← works if all robots have the same level

$i = 10, 11$

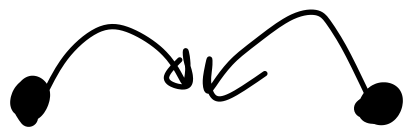
← works if Δ level ≤ 2

$i = 100, 101, 102$

What we would like to do

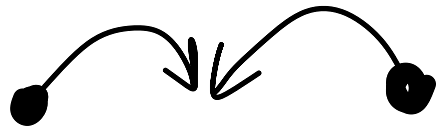
- If all the robots are correct we want all of them to execute move to middle

$i = 1$



← works if all robots have the same level

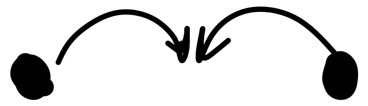
$i = 10, 11$



← works if $\Delta_{level} \leq 2$

$\Delta_{level} \leq 3$

$i = 100, 101, 102$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

$i = 1$ ← works if all robots have the same level

$i = 10, 11$ ← works if $\Delta_{level} \leq 2$

$\Delta_{level} \leq 3$

$i = 100, 101, 102$

$\vdots$

What we would like to do

- If all the robots are correct we want all of them to execute move to middle

$i = 1$ ← works if all robots have the same level

← make sure level increases

$i = 10, 11$ ← works if $\Delta_{level} \leq 2$

$\Delta_{level} \leq 3$

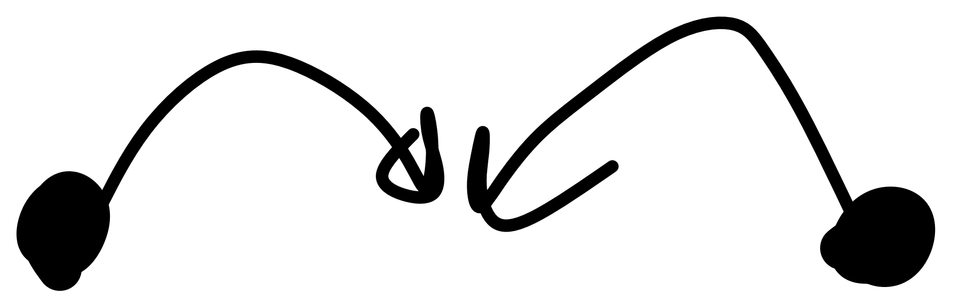
$i = 100, 101, 102$

⋮

What we would like to do

- If all the robots are correct we want all of them to execute move to middle

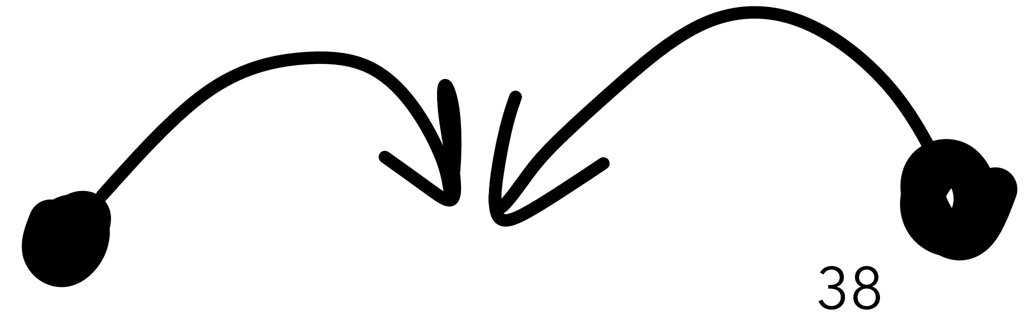
$i = 1$



← wor

← make sure

$i = 10, 11$

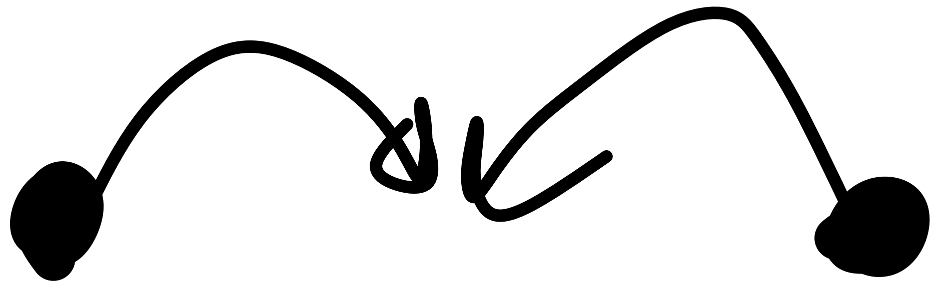


← wo

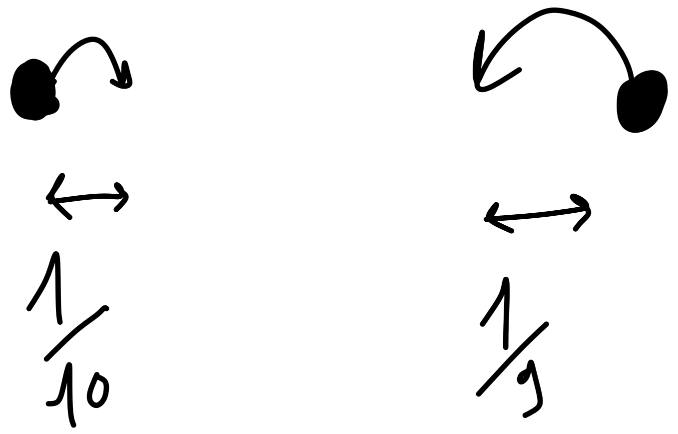
What we would like to do

- If all the robots are correct we want all of them to execute move to middle

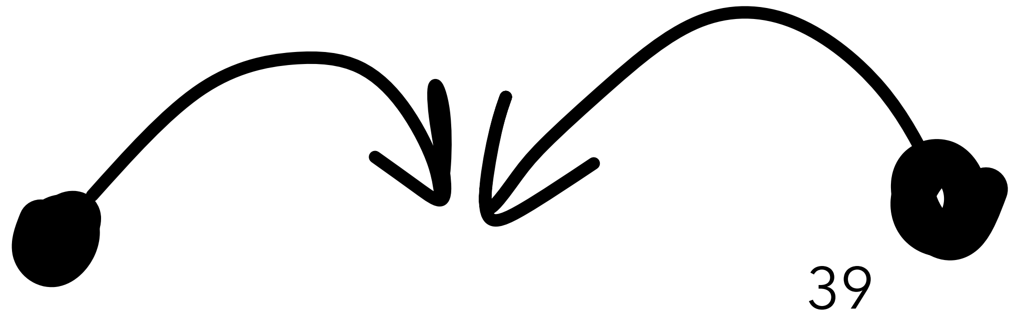
$i = 1$



$i = 2$



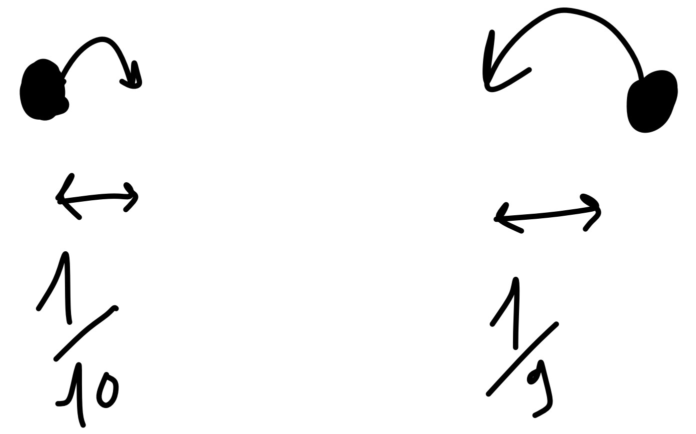
$i = 10, 11$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

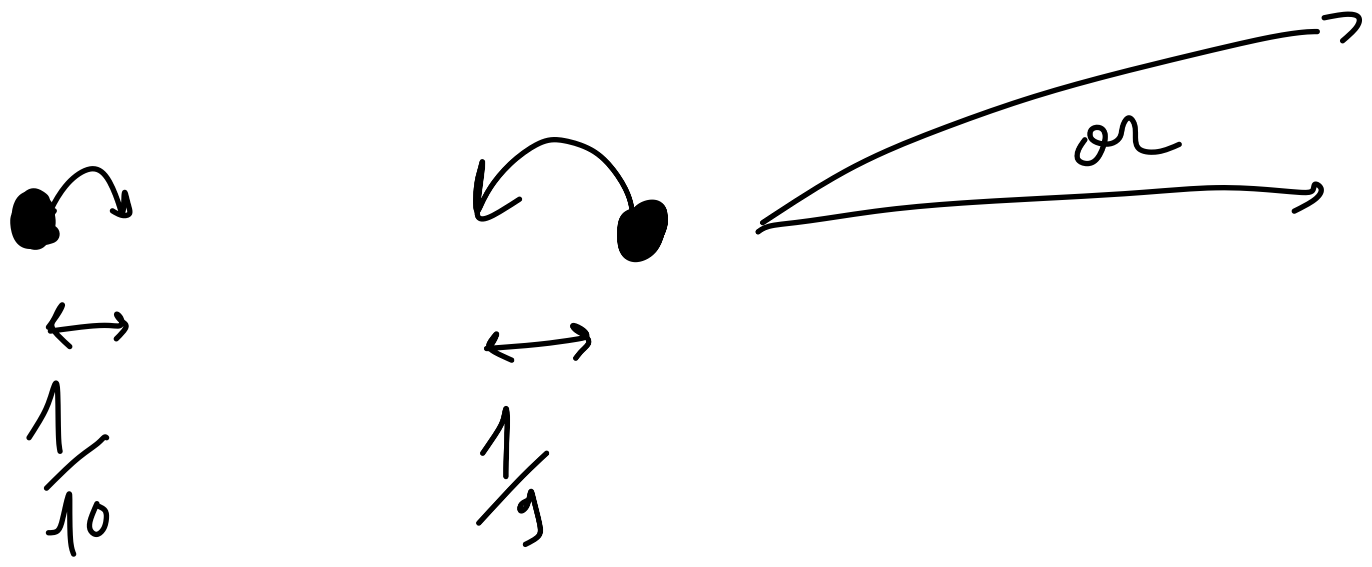
$i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

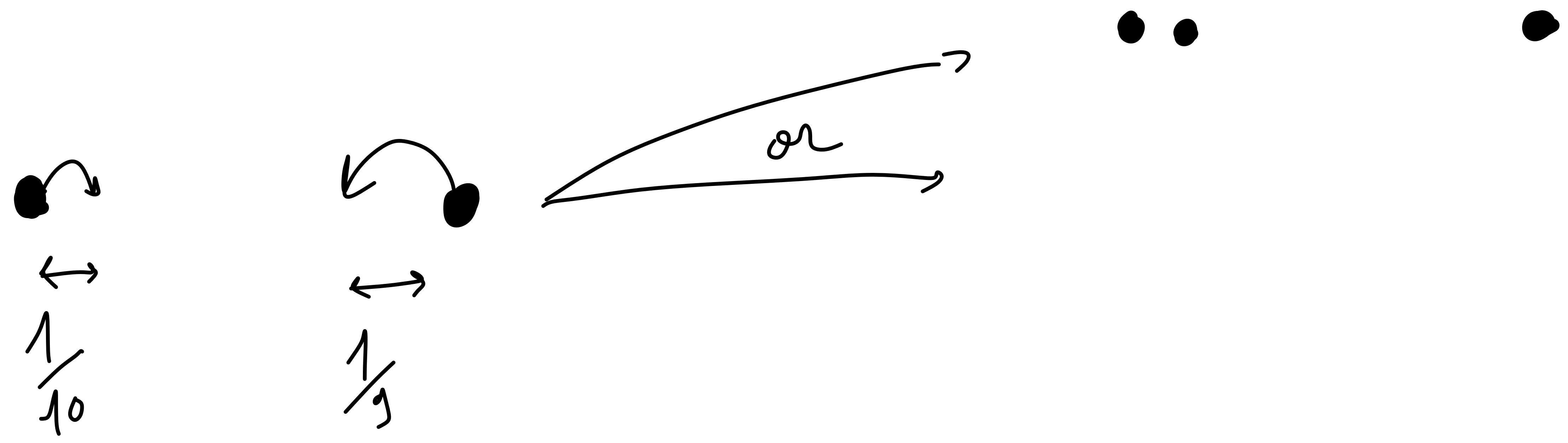
$($ $i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

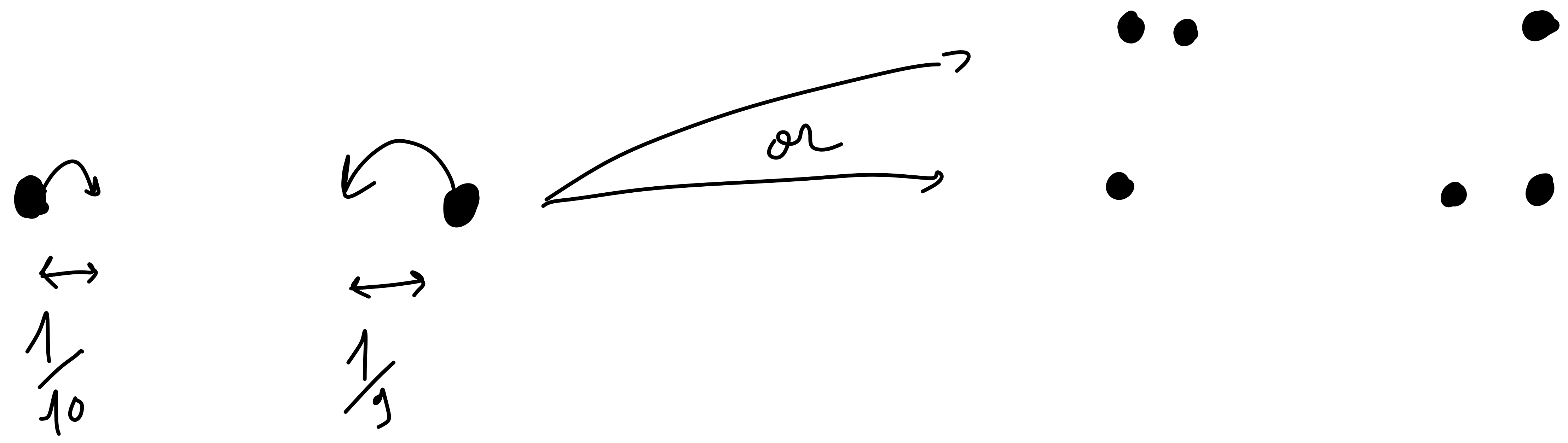
$i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

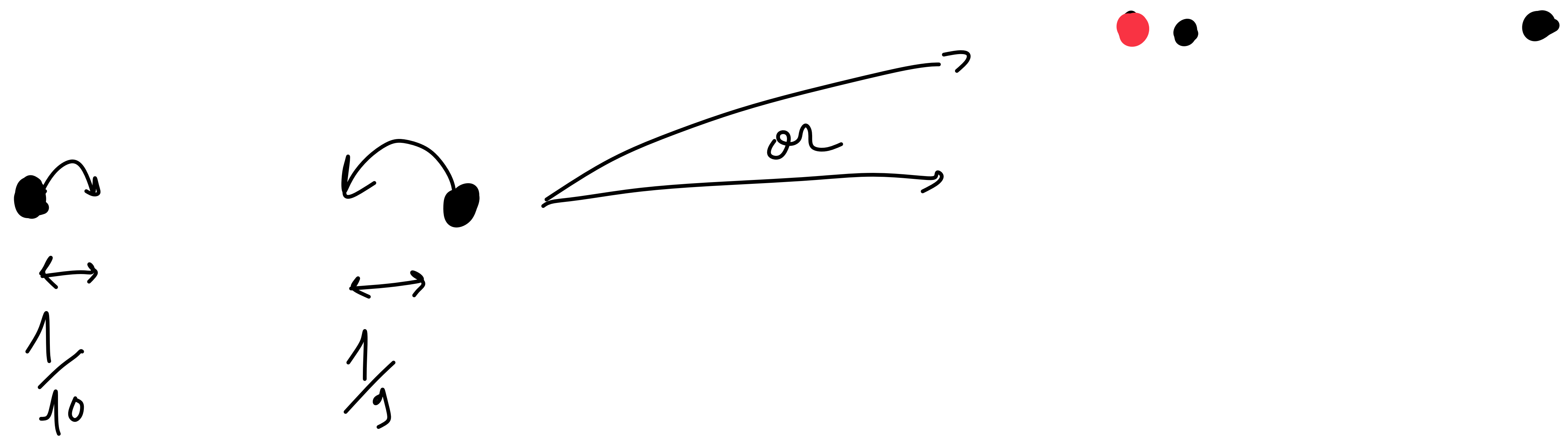
$i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

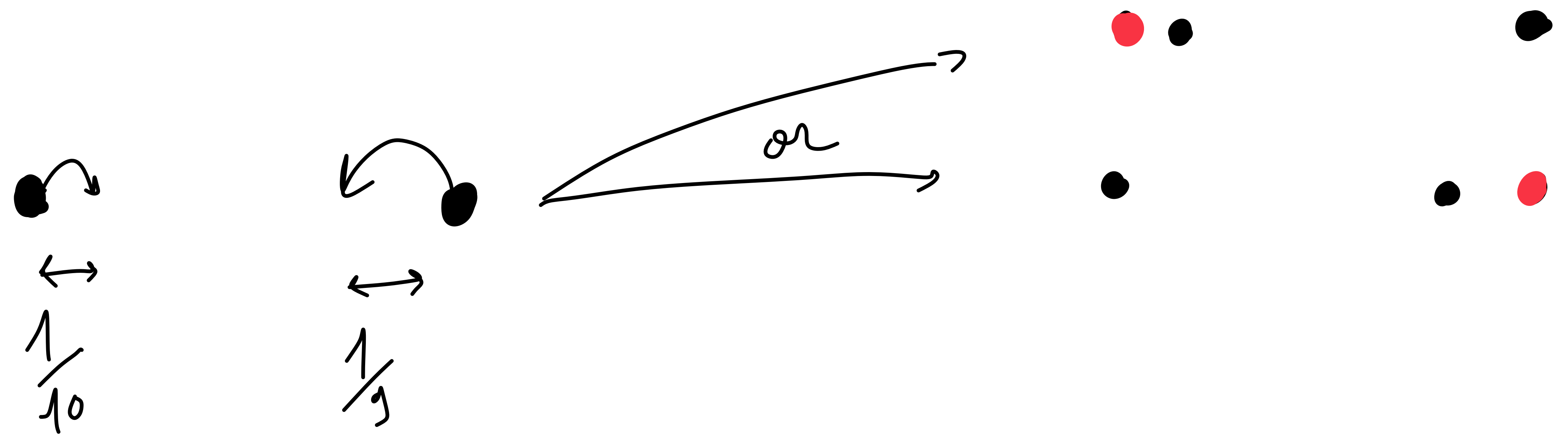
$i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

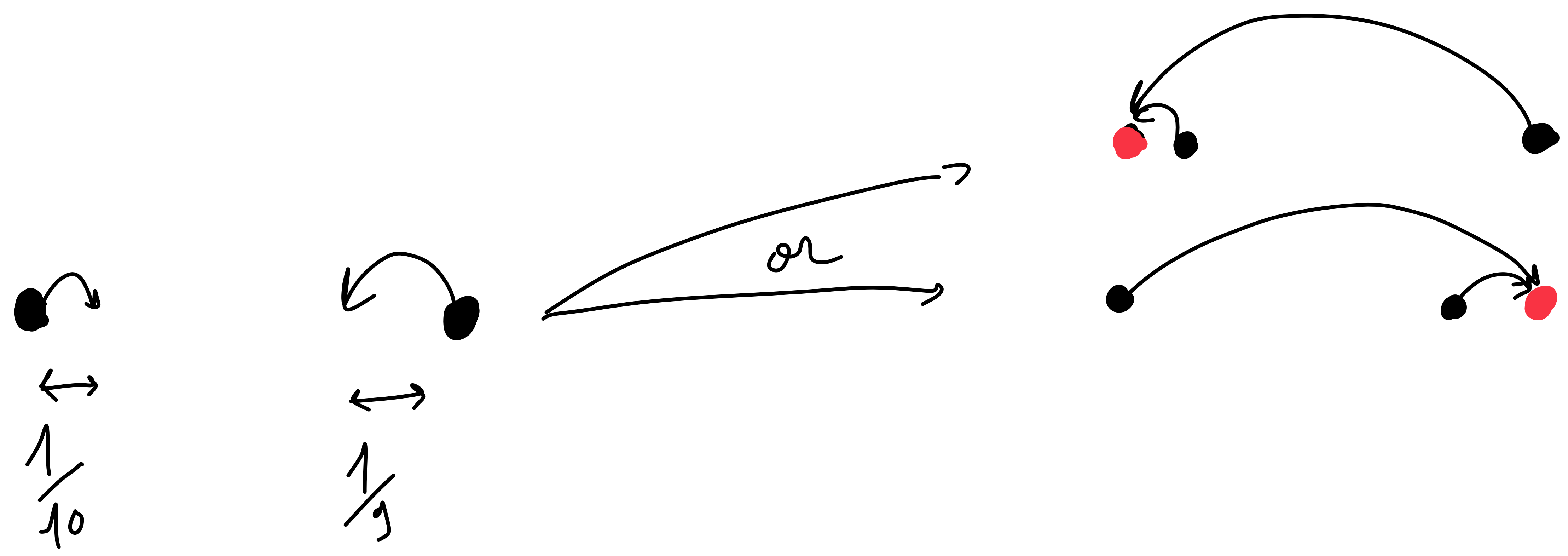
$i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

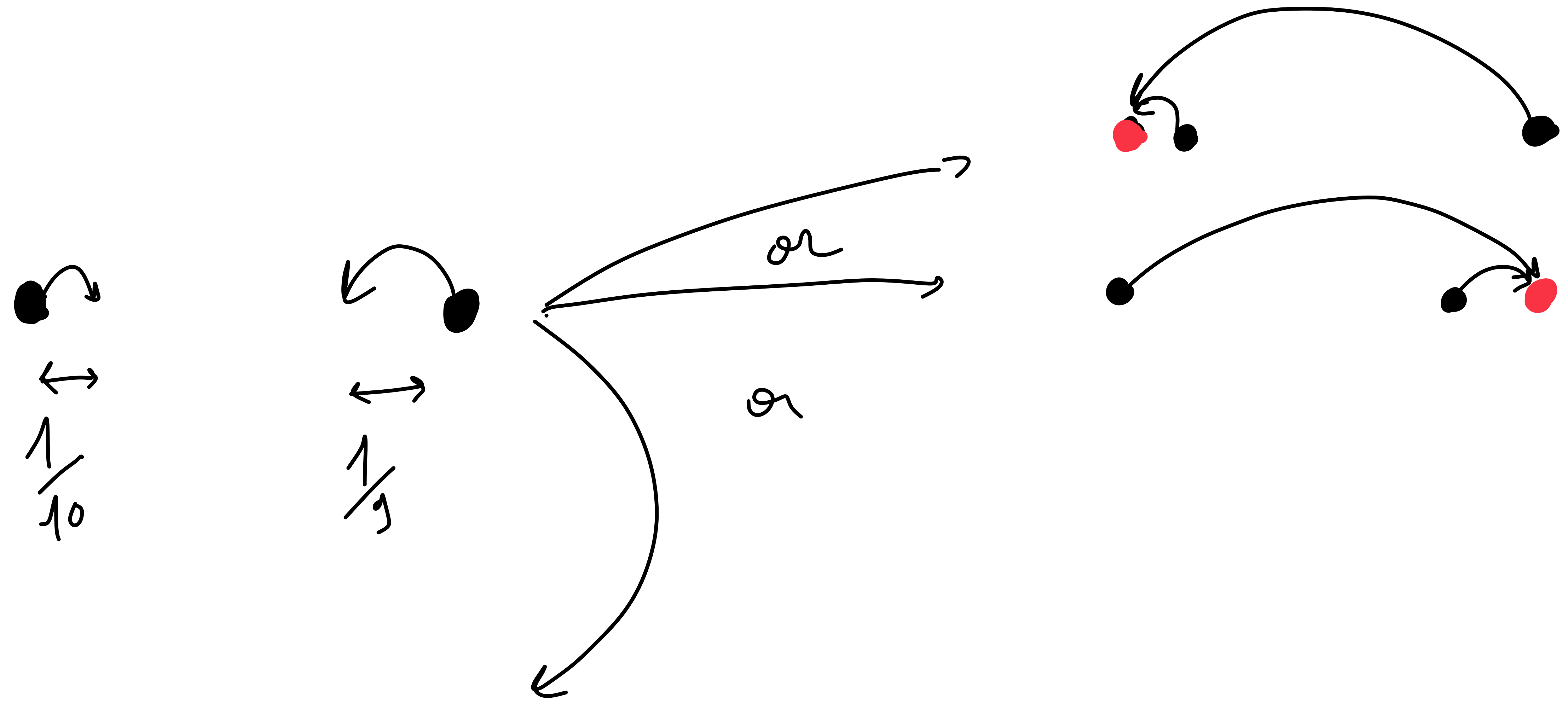
$i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

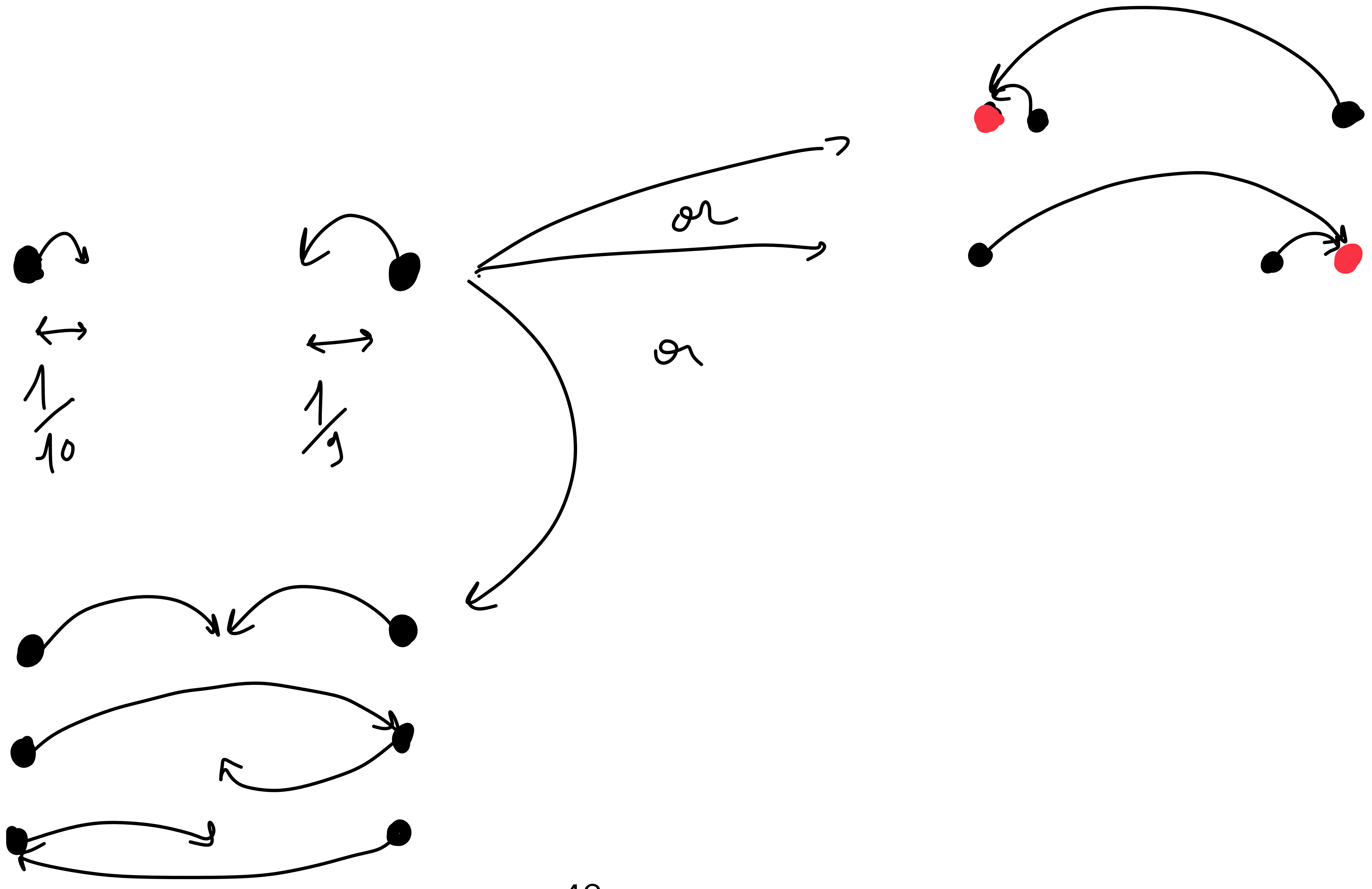
$i = 2$



What we would like to do

- If all the robots are correct we want all of them to execute move to middle

i = 2



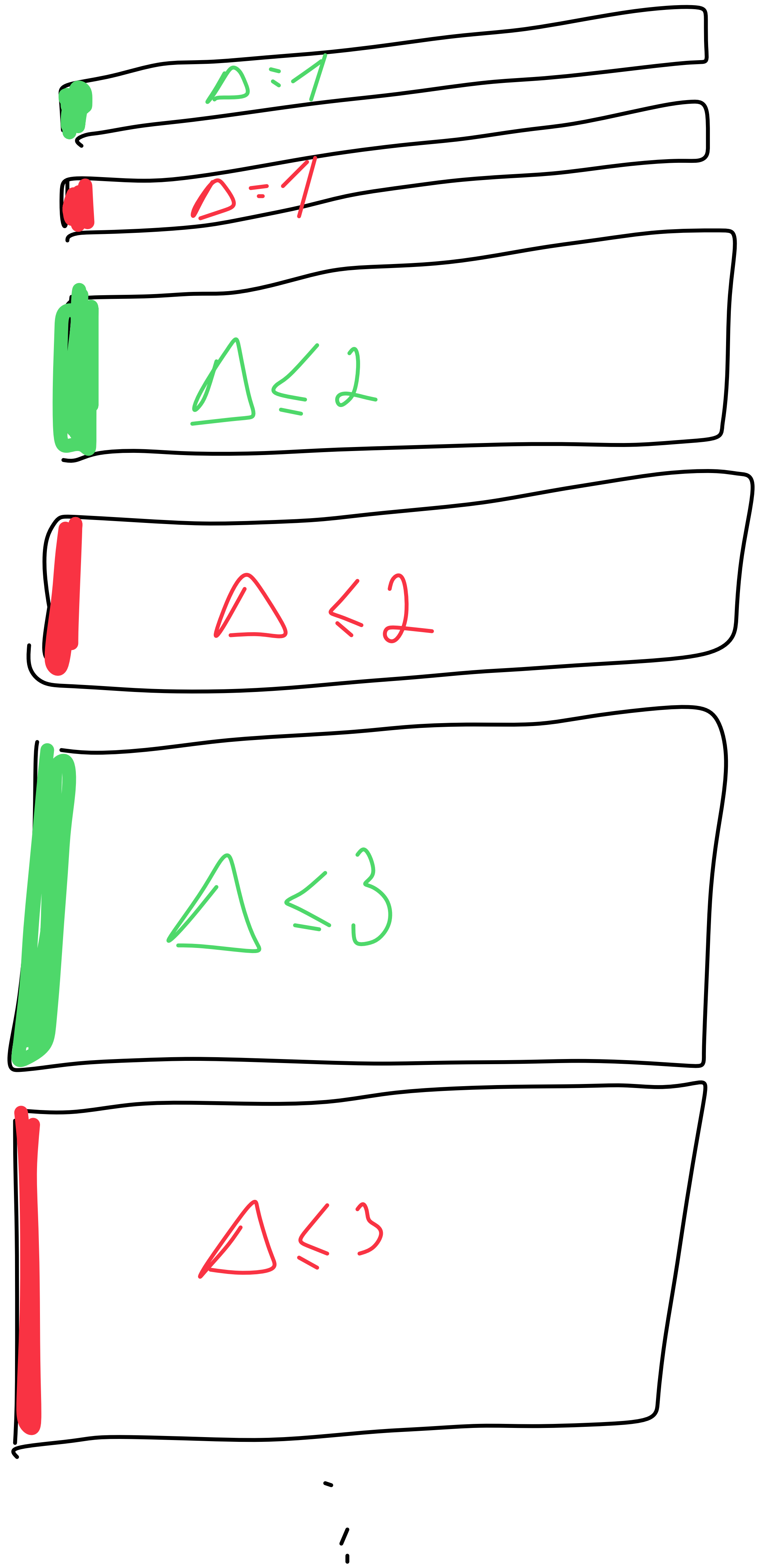
i = 3

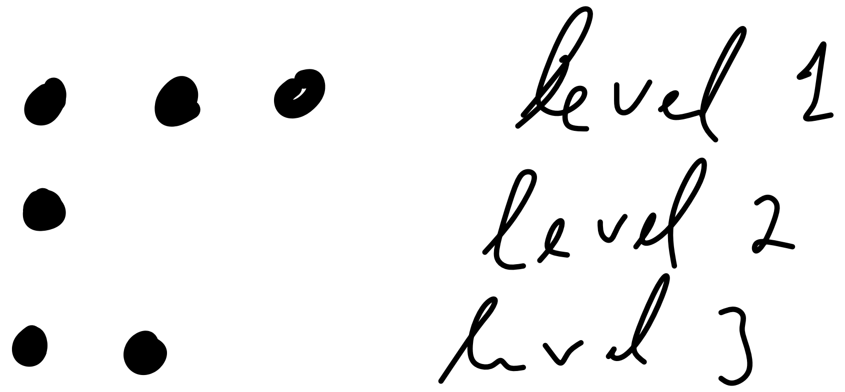
i = 4

i = 5

The Level-Slicing Technic

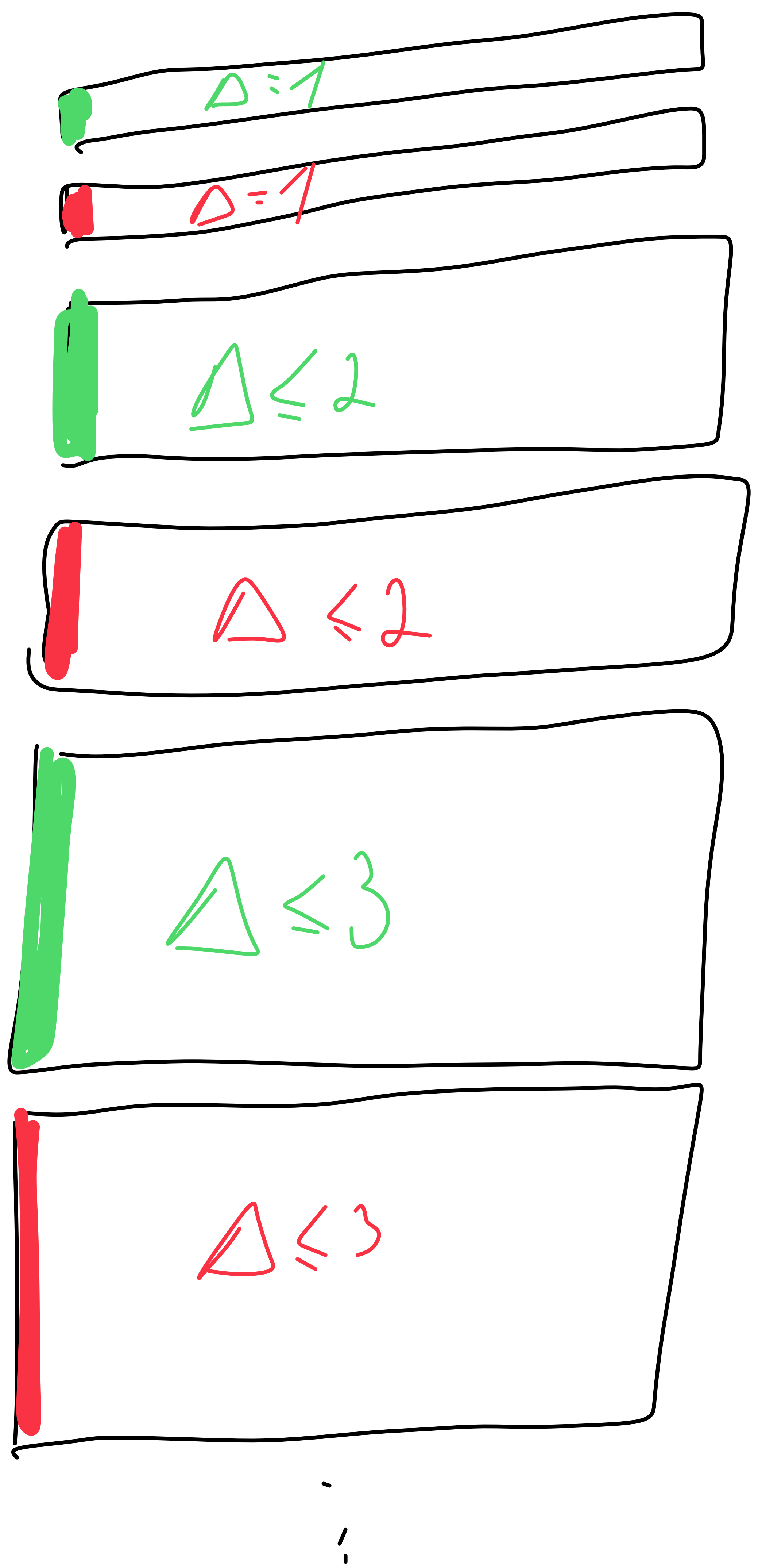
level i

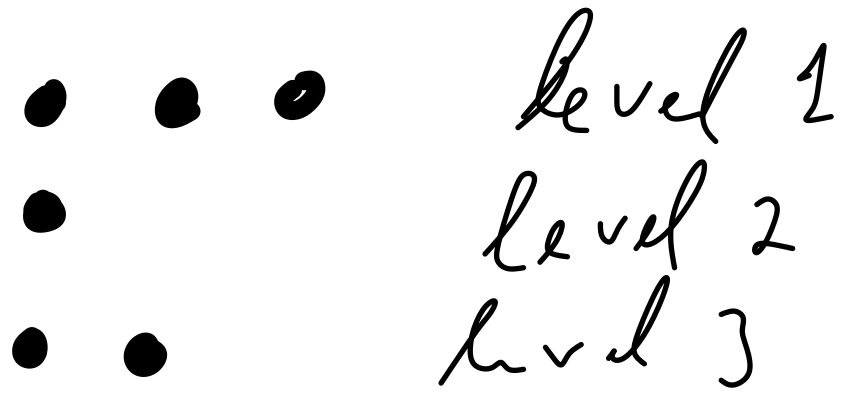




The Level-Slicing Technic

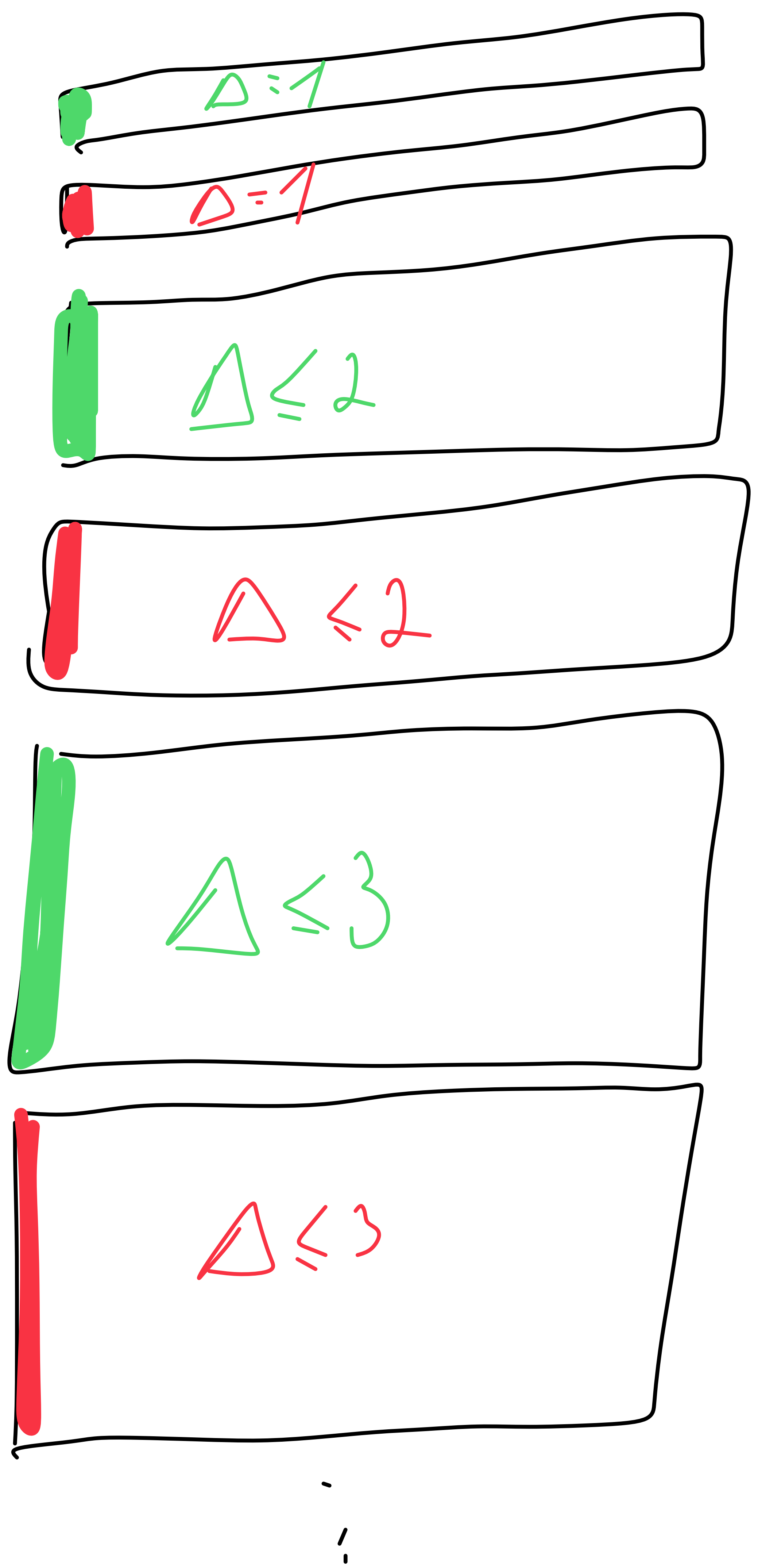
level i

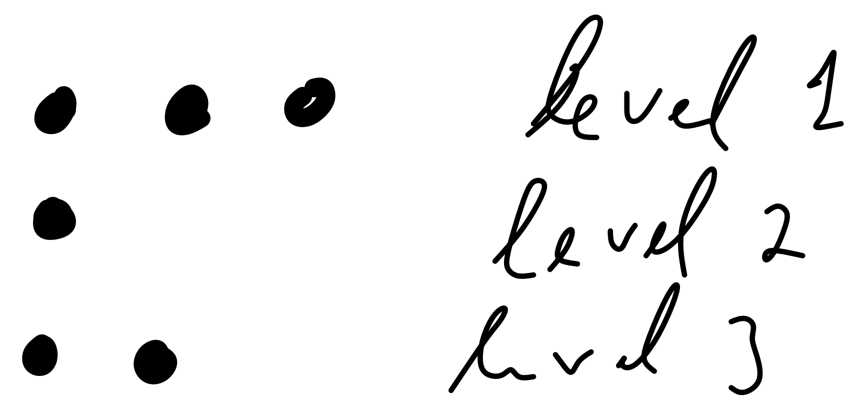




The Level-Slicing Technic

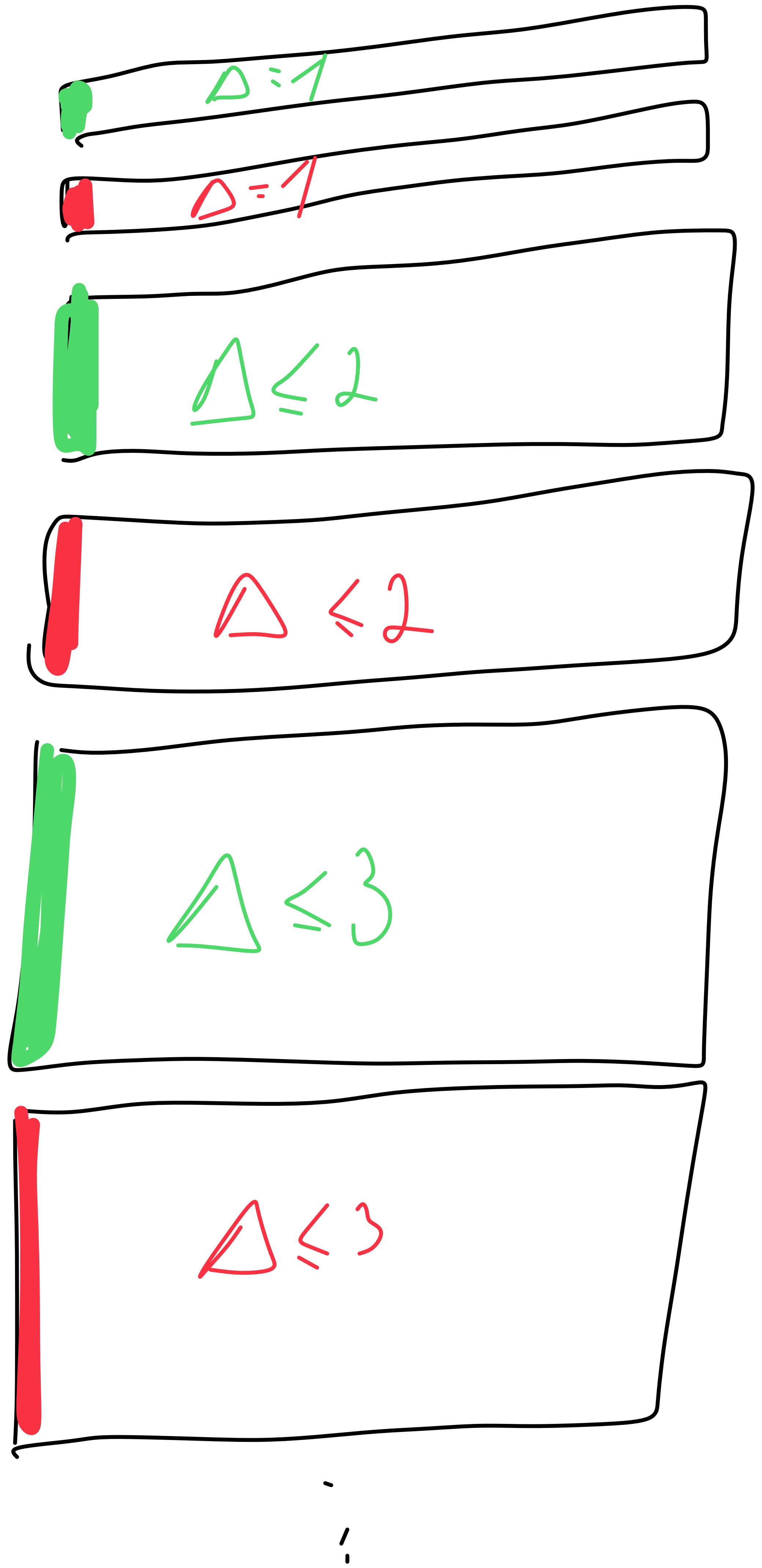
level i

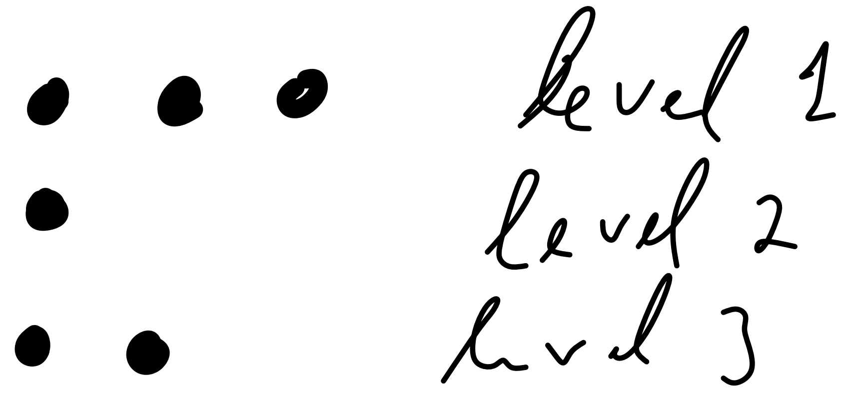




The Level-Slicing Technic

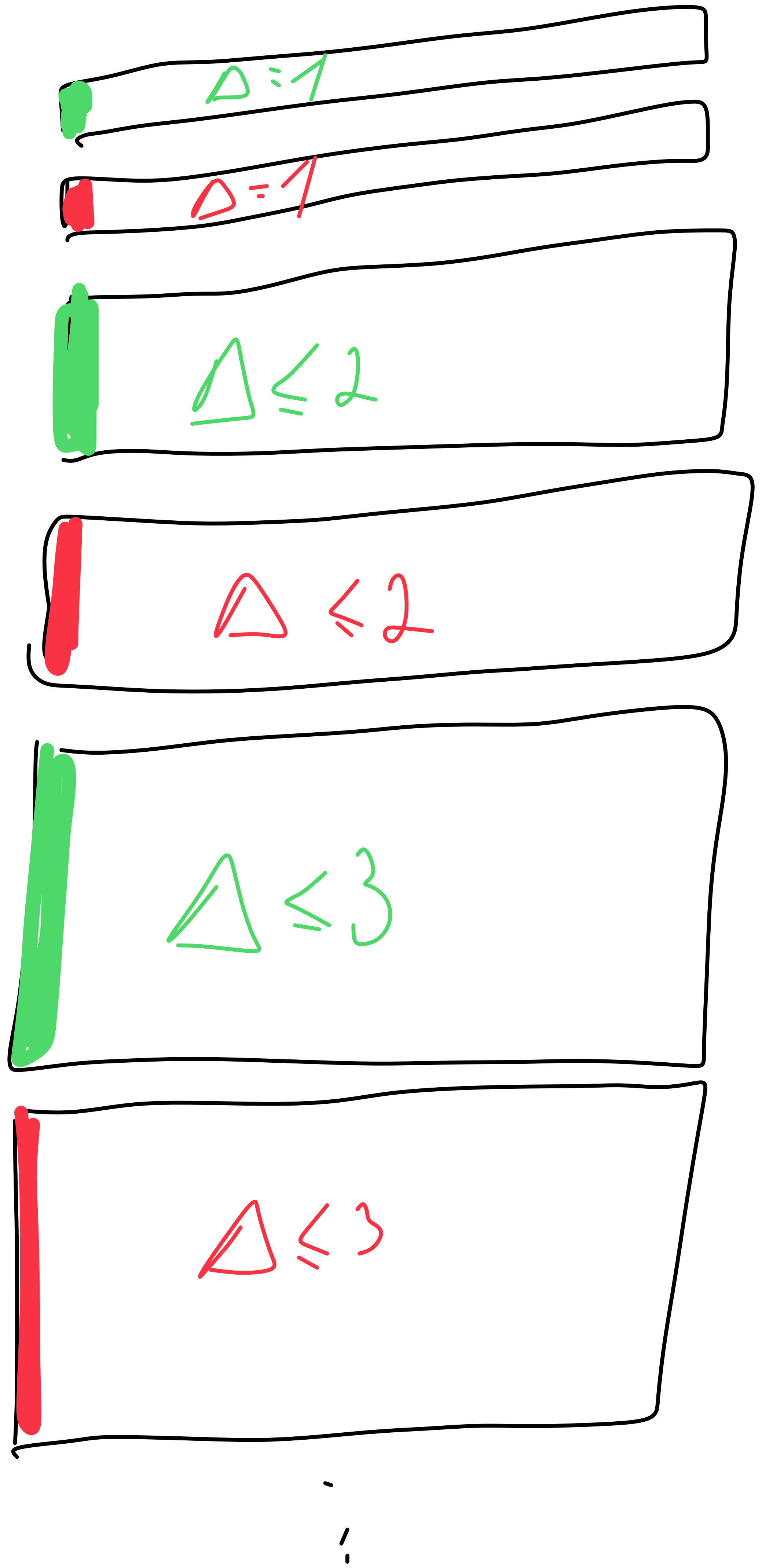
level i

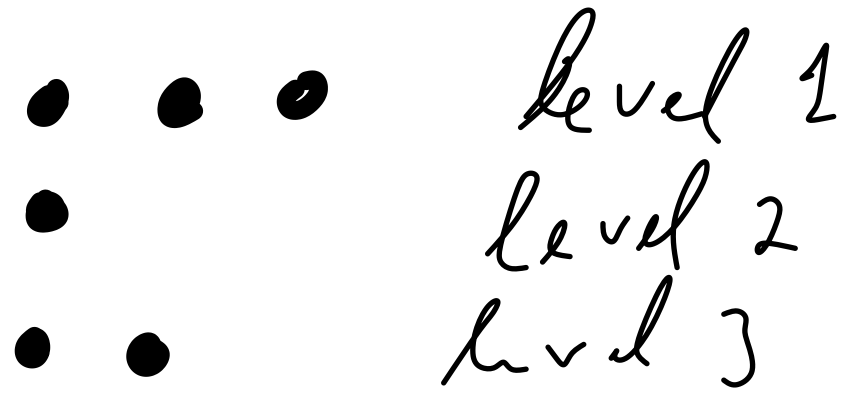




The Level-Slicing Technic

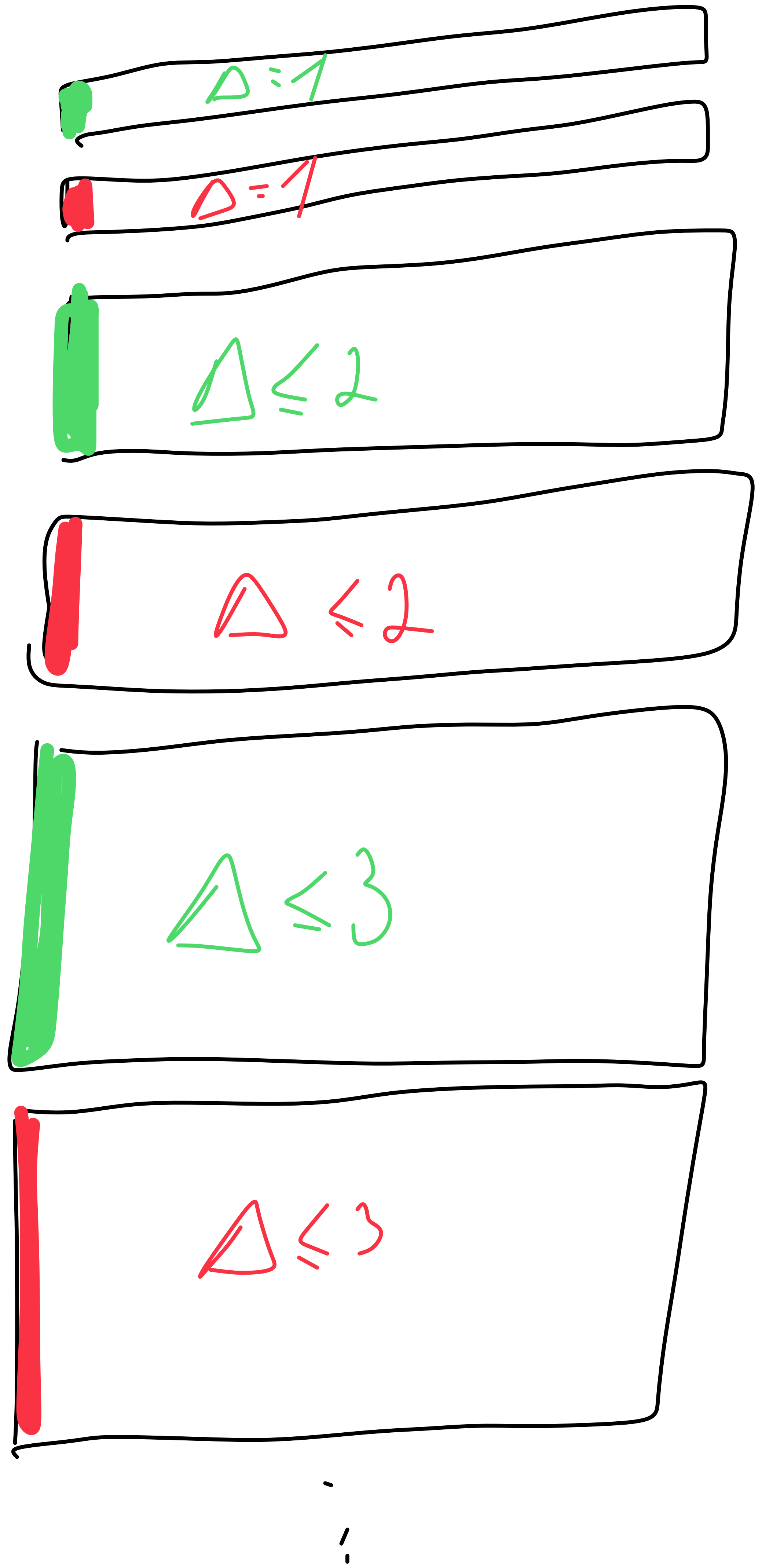
level i

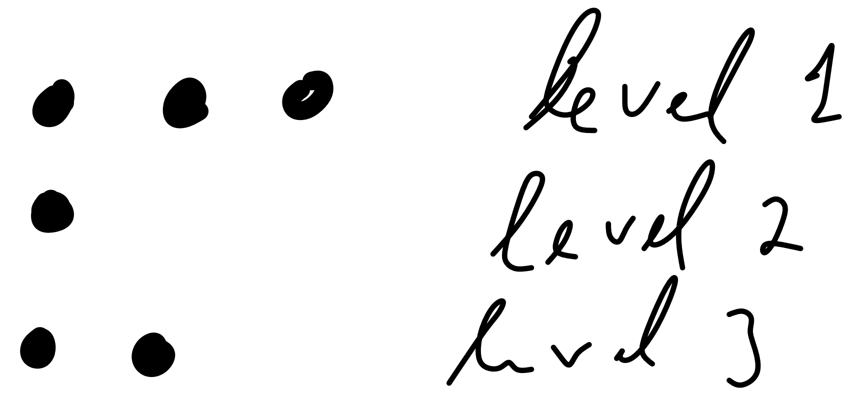




The Level-Slicing Technic

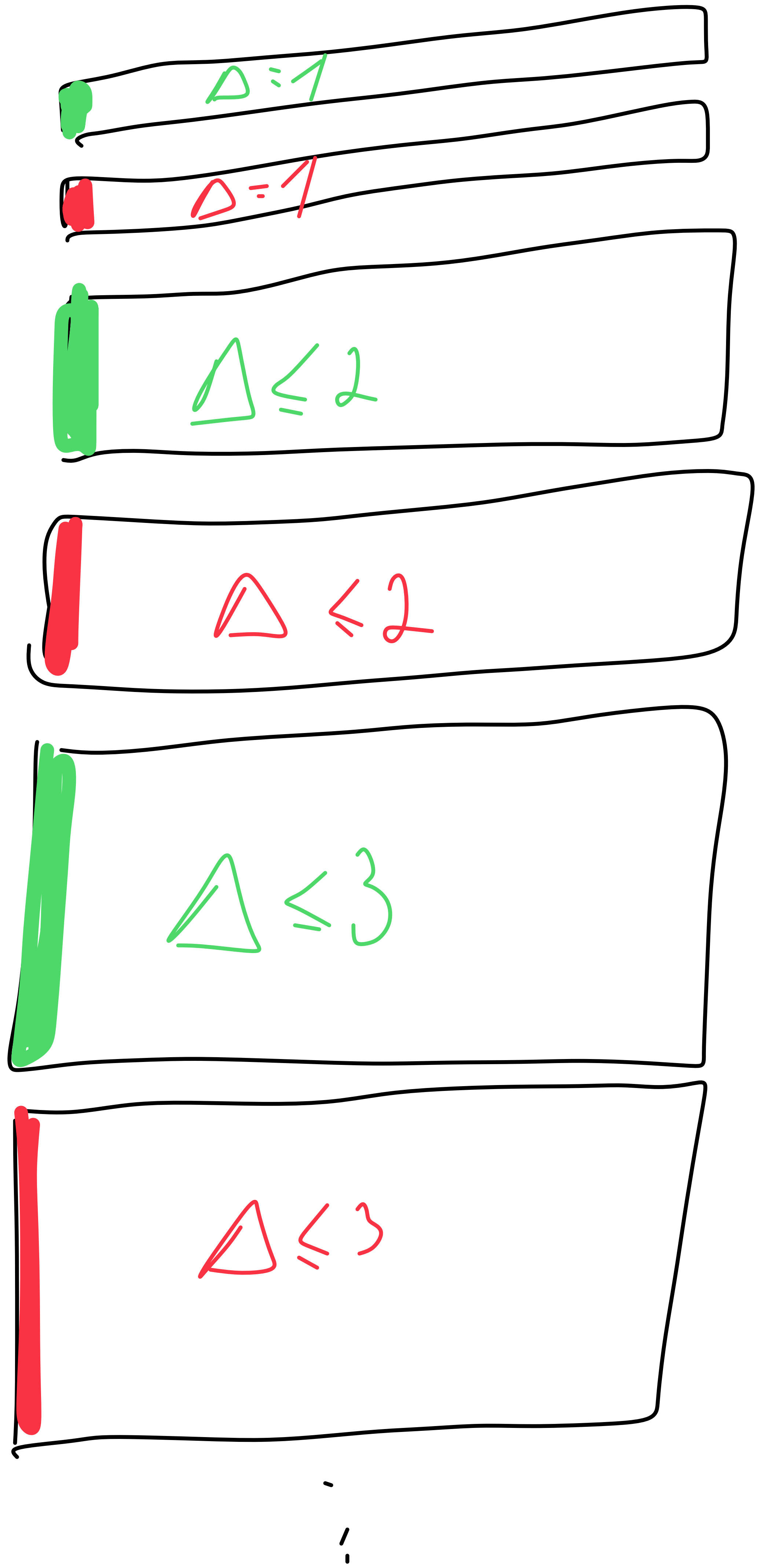
level i

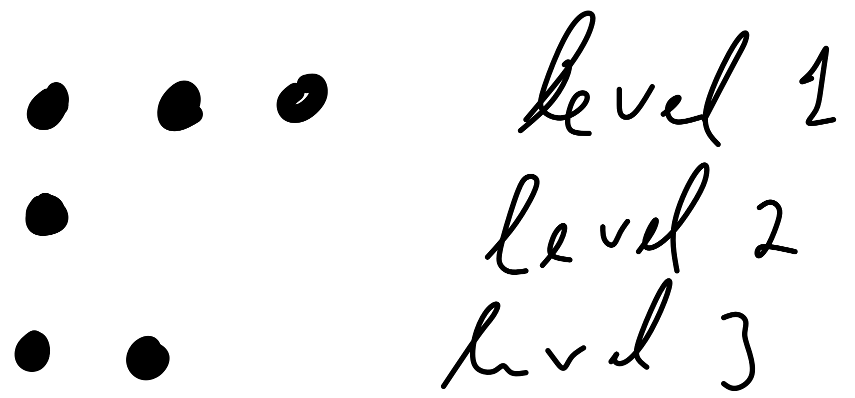




The Level-Slicing Technic

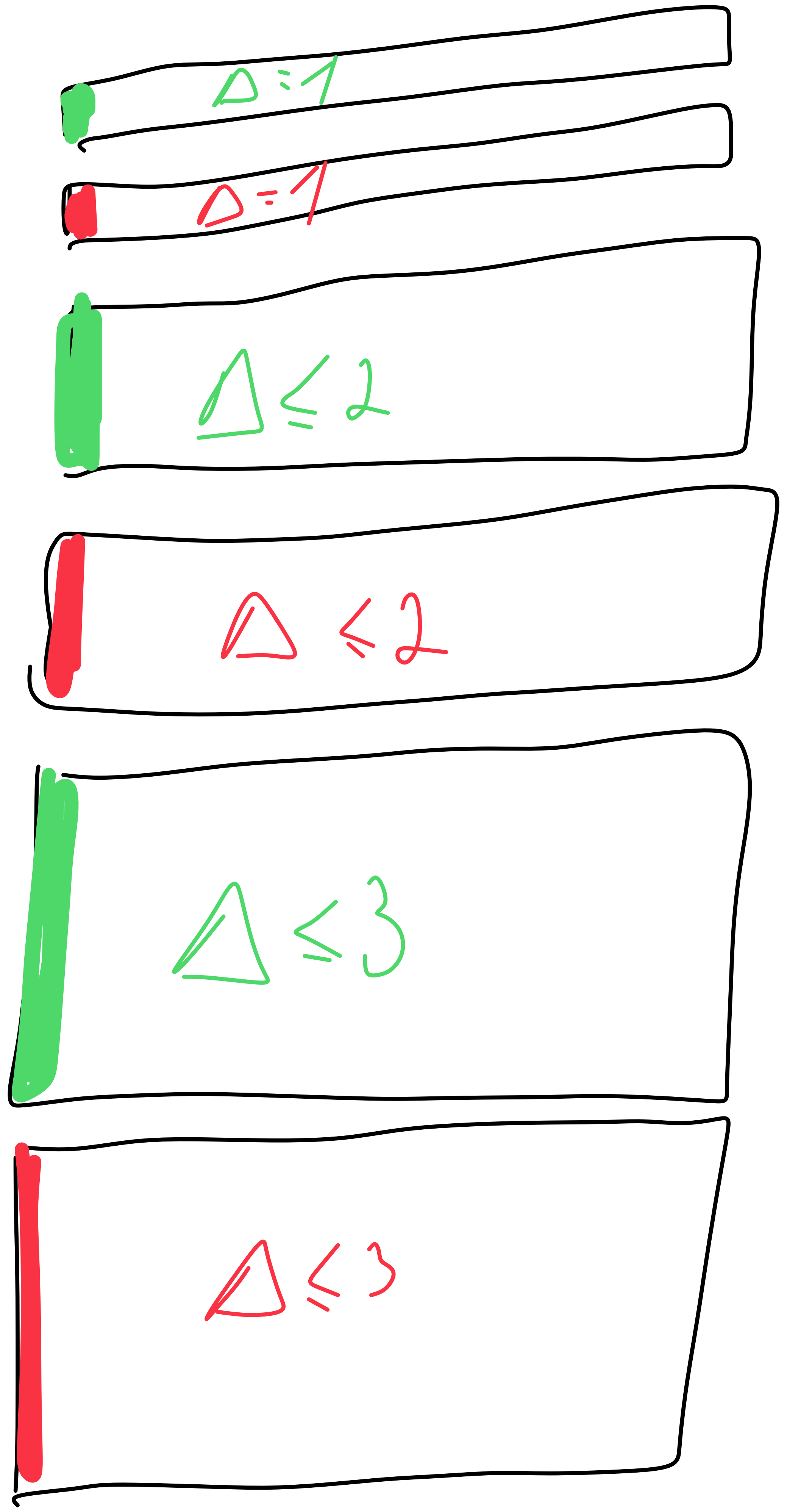
level i

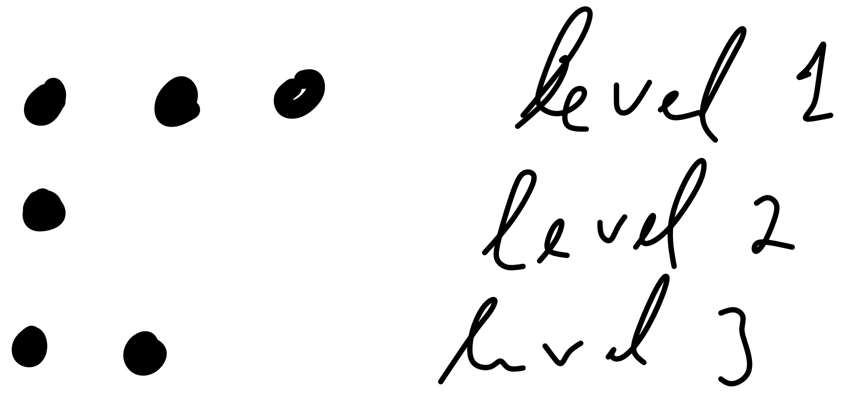




The Level-Slicing Technic

level i





Conclusion

Conclusion

- Solving strong (crash-tolerant) gathering in FSYNC without any additional assumption on the robots nor on the initial configuration

Conclusion

- Solving strong (crash-tolerant) gathering in FSYNC without any additional assumption on the robots nor on the initial configuration
- New level-slicing technic that can be of independent interest

Conclusion

- Solving strong (crash-tolerant) gathering in FSYNC without any additional assumption on the robots nor on the initial configuration
- New level-slicing technic that can be of independent interest

Conclusion

- Solving strong (crash-tolerant) gathering in FSYNC without any additional assumption on the robots nor on the initial configuration
- New level-slicing technic that can be of independent interest

Conclusion

- Solving strong (crash-tolerant) gathering in FSYNC without any additional assumption on the robots nor on the initial configuration
- New level-slicing technic that can be of independent interest

- What assumption can render the problem solvable in SSYNC?

Conclusion

- Solving strong (crash-tolerant) gathering in FSYNC without any additional assumption on the robots nor on the initial configuration
- New level-slicing technic that can be of independent interest

- What assumption can render the problem solvable in SSYNC?

Thank you!