Battleship Complexity

GT GDMM 2021

March 16th 2021, Nancy, France

Yan Gerard

Joined work with Loïc Crombez and Guilherme Da Fonseca

UNIVERSITÉ Clermont Auvergne

LIMOS

Plan

I Problem Statement

II Shooting Algorithms

III Results

IV Proofs

Plan

I Problem Statement

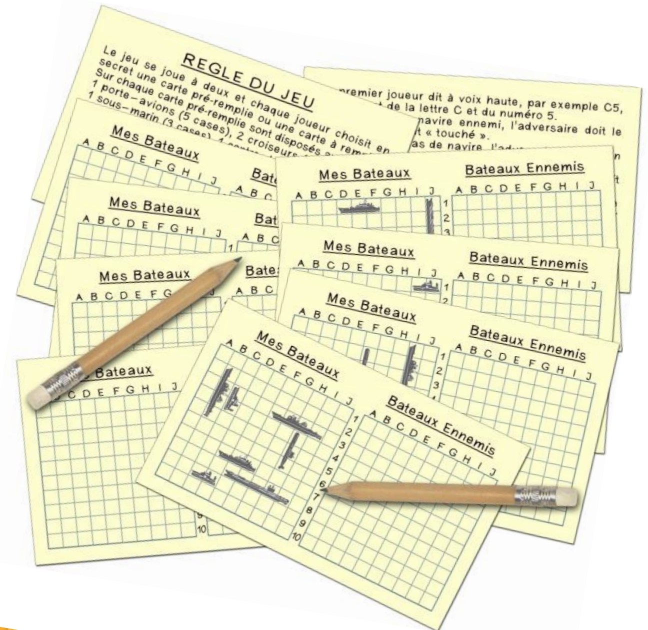
II Shooting Algorithms

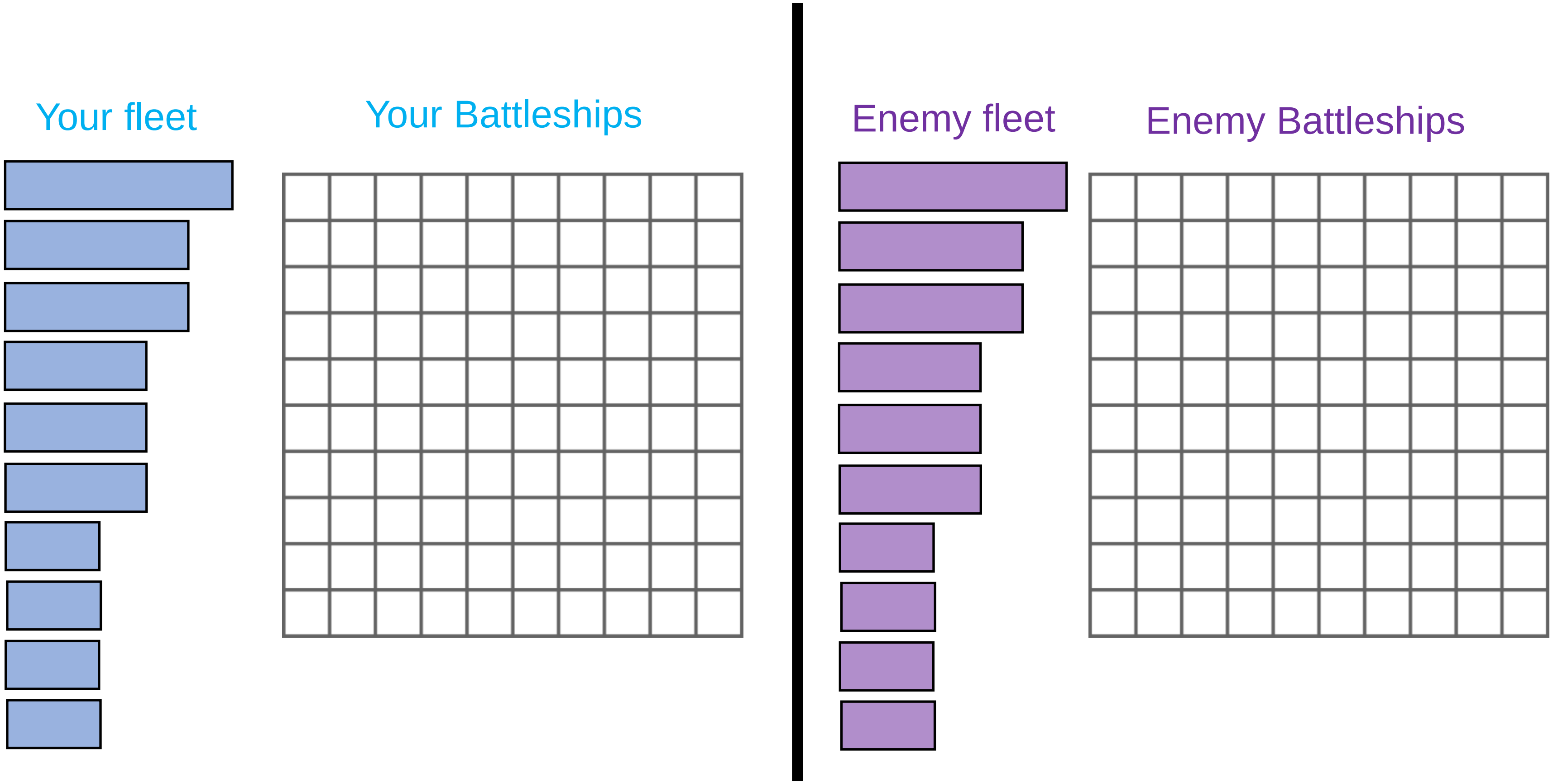
III Results

IV Proofs

1 Problem statement

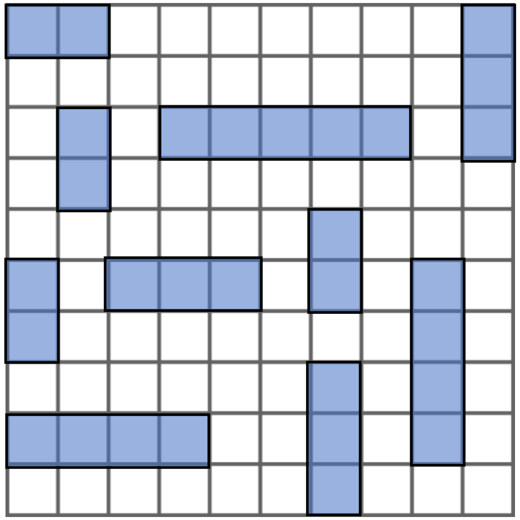
The game



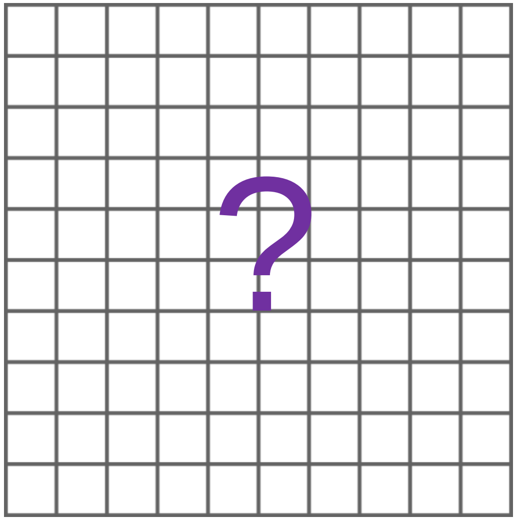


2 players : you and the enemy

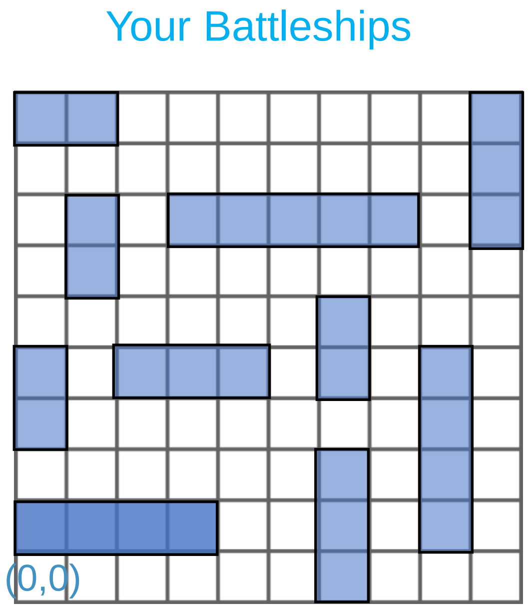
Your Battleships

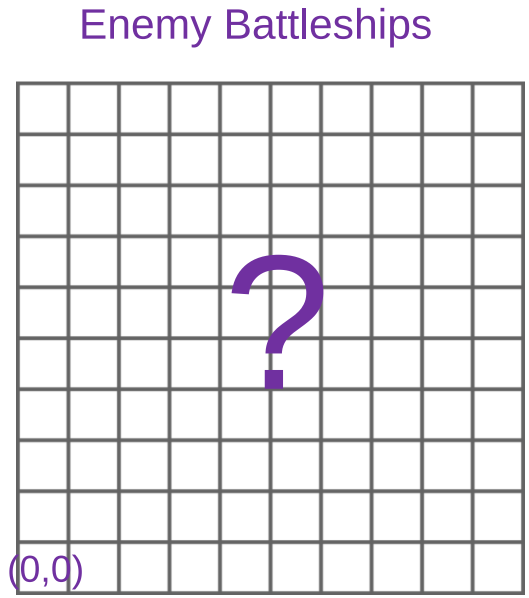


Enemy Battleships

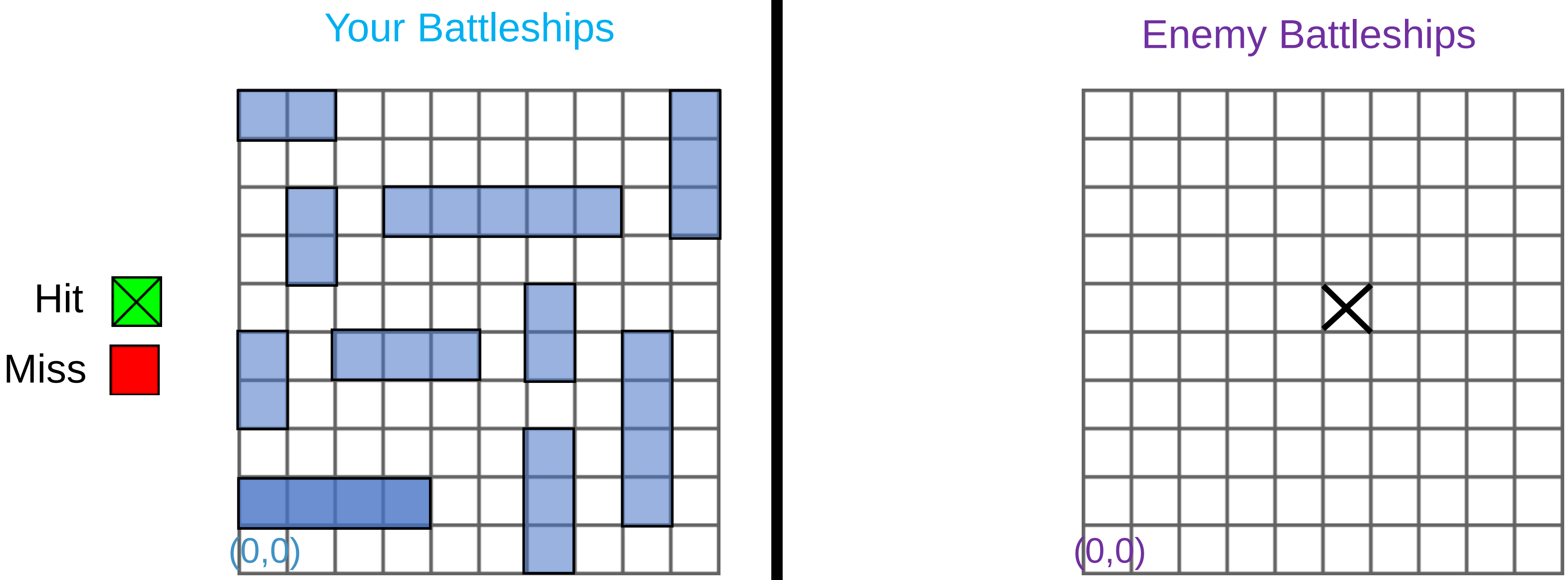


2 players : you and the enemy

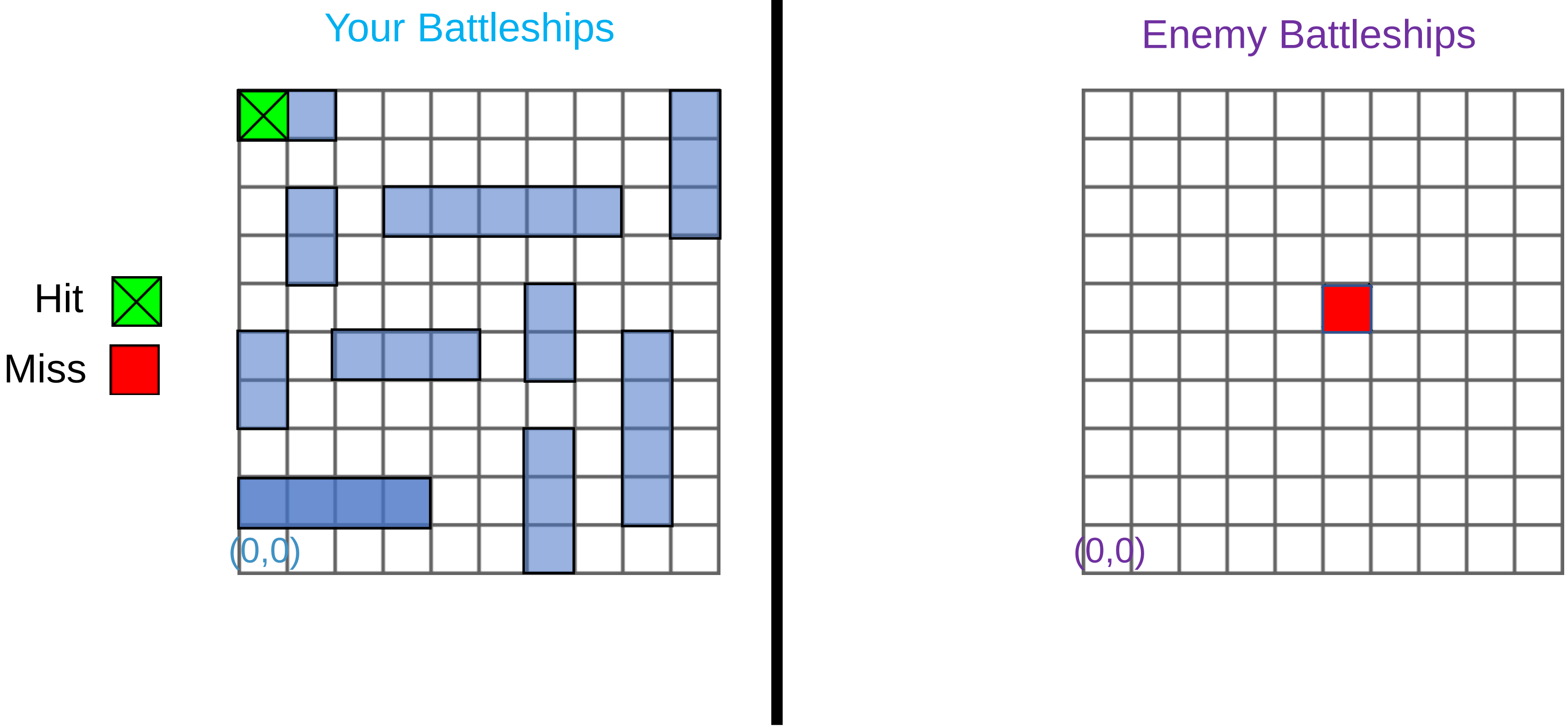




At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

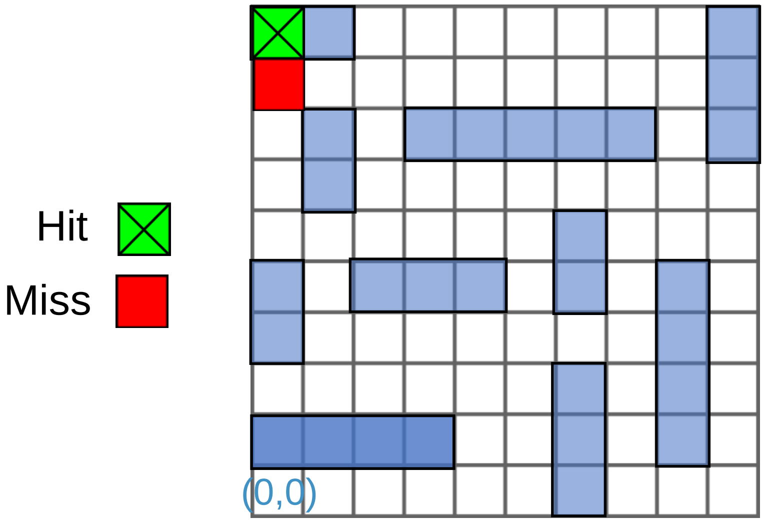


At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

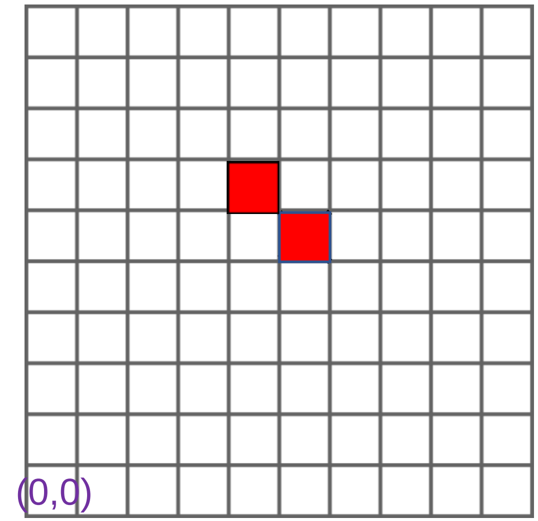


At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

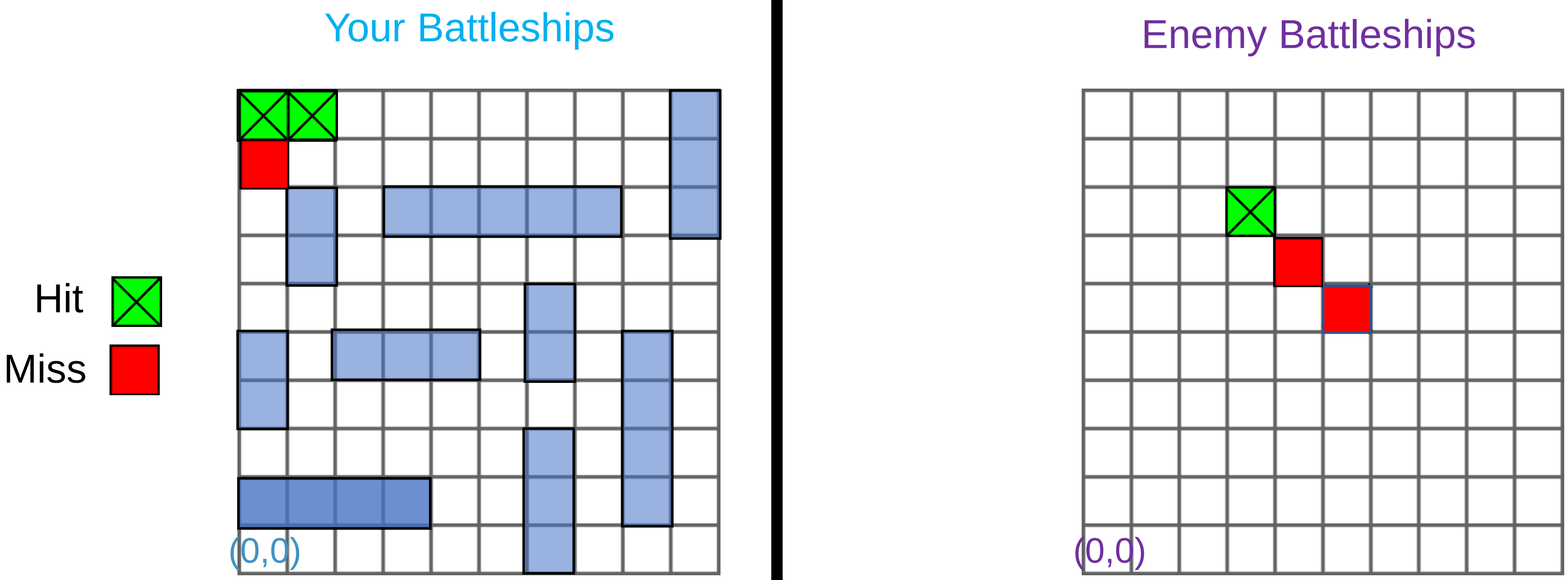
Your Battleships



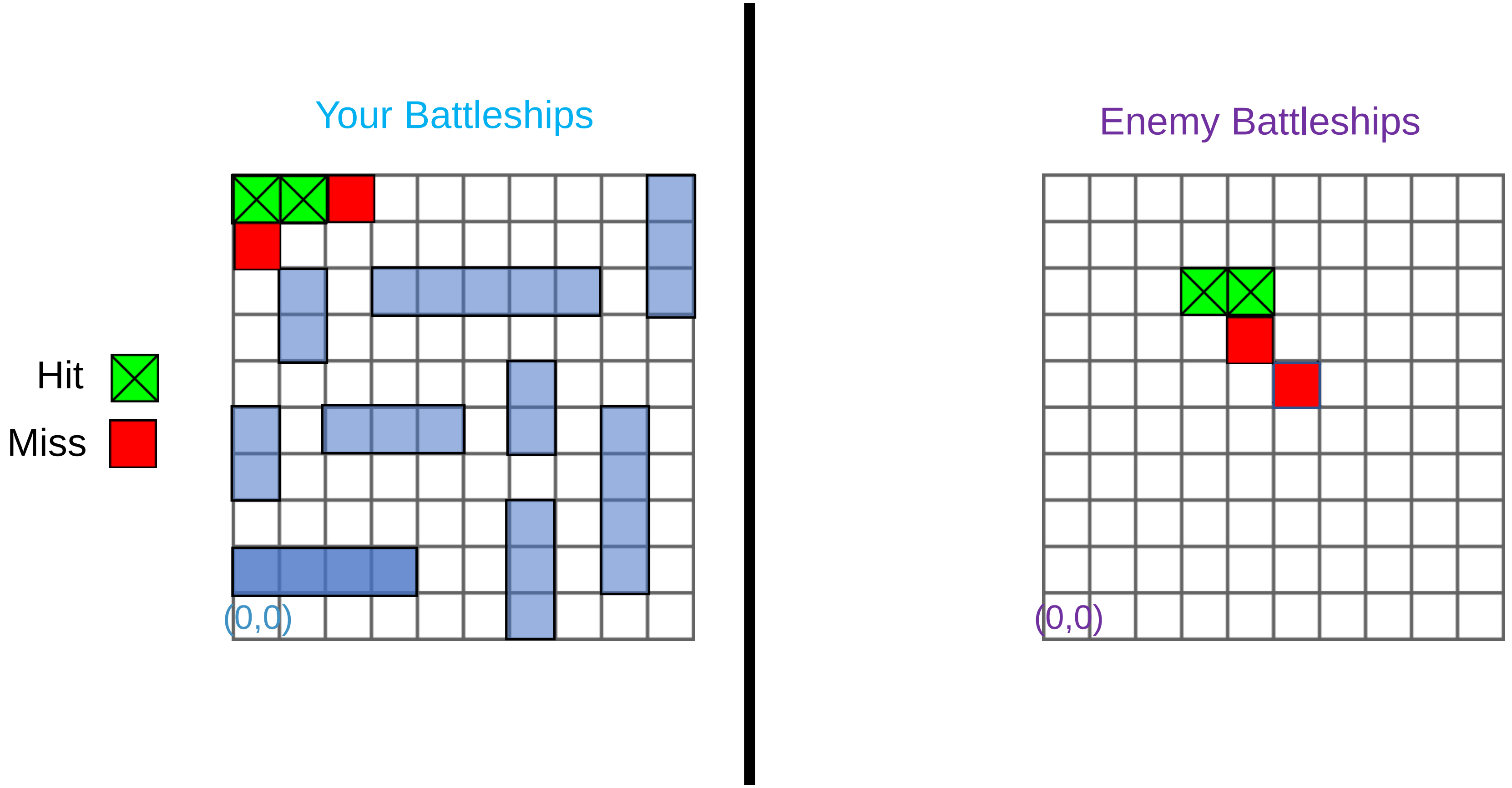
Enemy Battleships



At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss



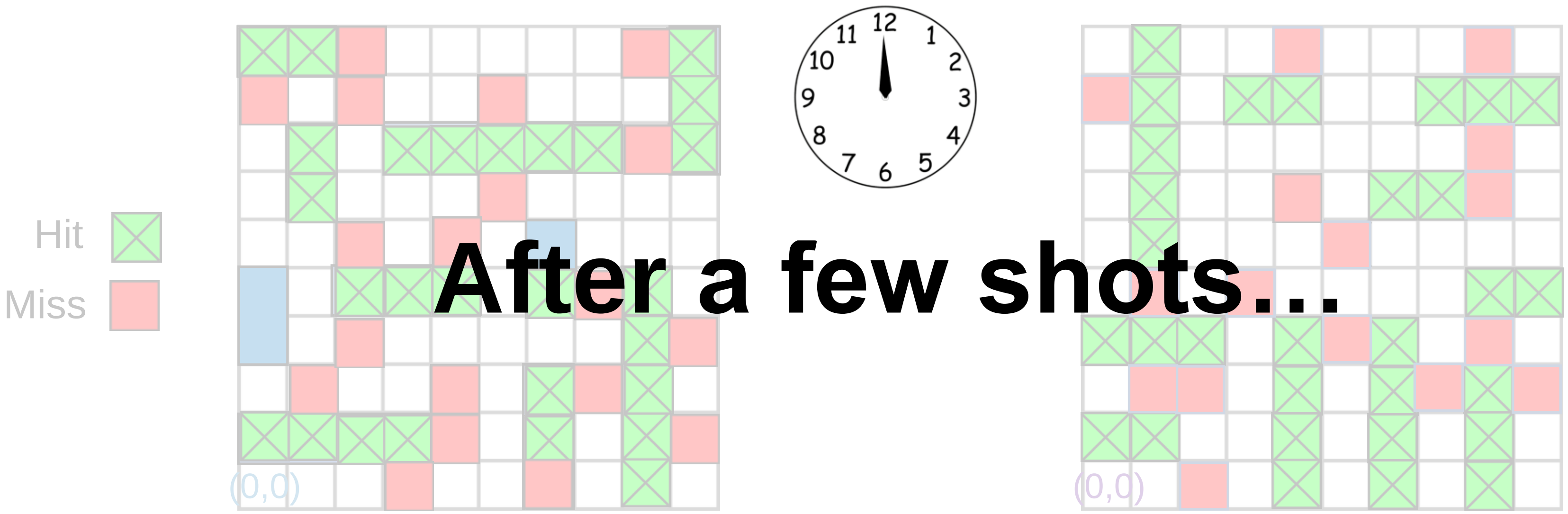
At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss



At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

Your Battleships

Enemy Battleships

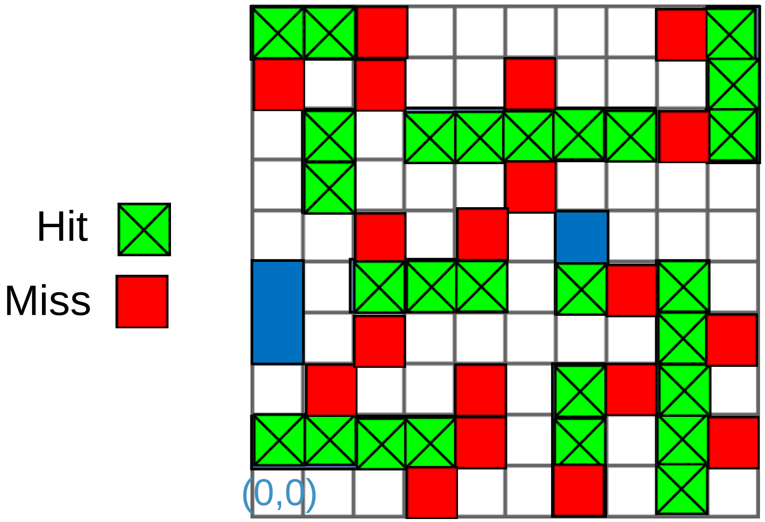


You still have safe squares

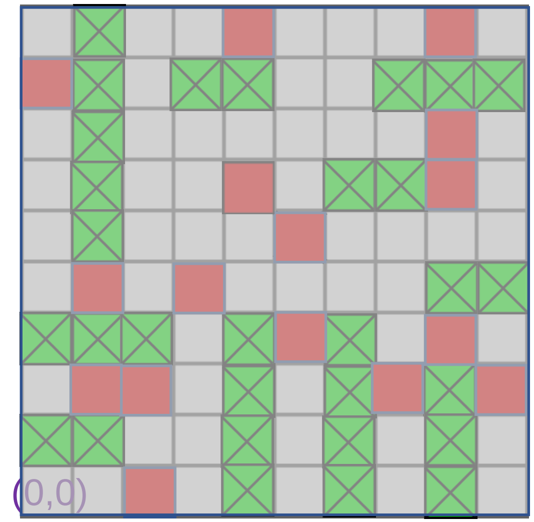
All enemy ships have been sunk

At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

Your Battleships



Enemy Battleships



You still have safe squares

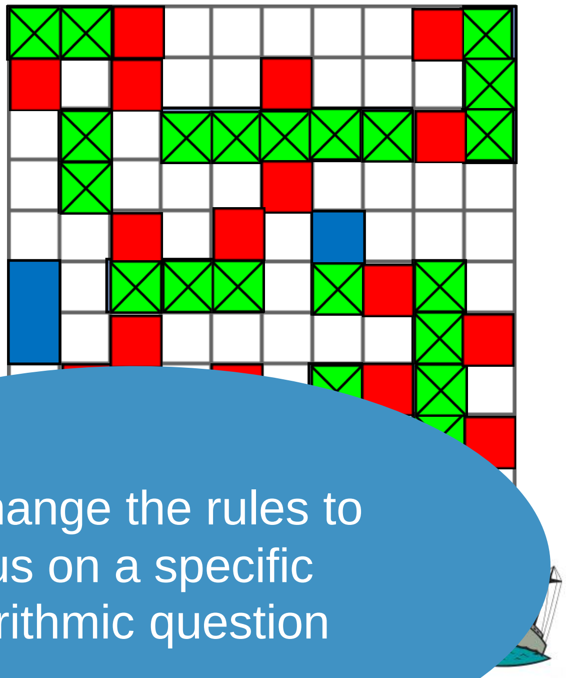
All enemy ships have been sunk

At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

The winner is the first player which has hit all the squares of the enemy fleet

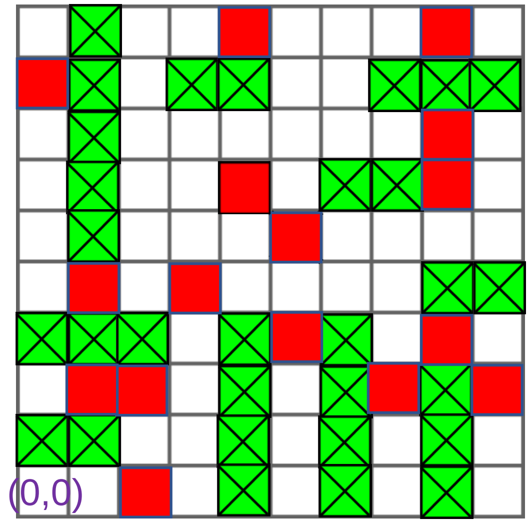
Your Battleships



Hit

Miss

Enemy Battleships



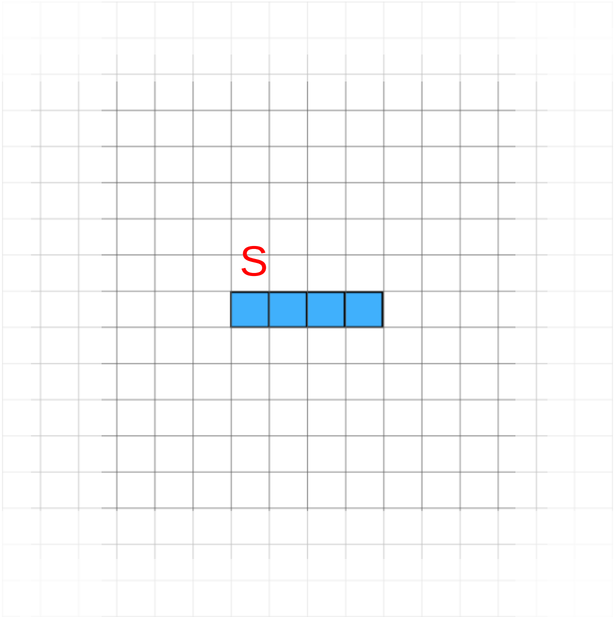
All enemy ships have been sunk

We change the rules to focus on a specific algorithmic question

At each turn, shoot a square in your enemy grid => the enemy tells you if it is a hit or a miss

The winner is the first player which has hit all the squares of the enemy's fleet

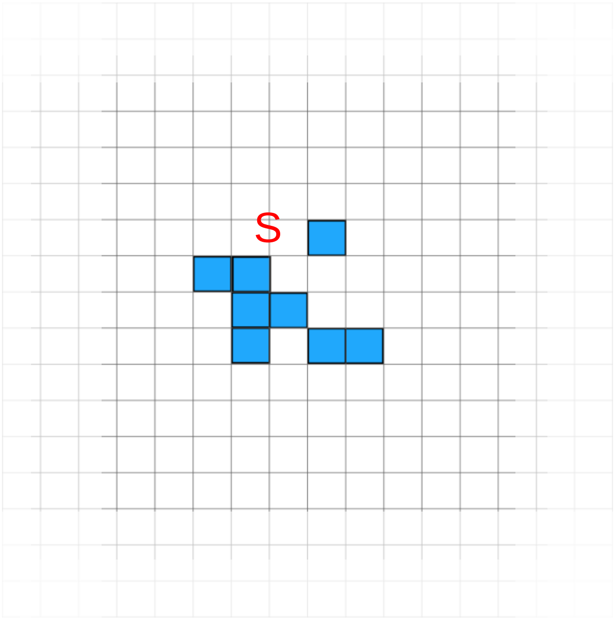
- Only one ship S in an unbounded grid (no information comes from a position close to the boundary of the grid)



We change the rules to focus on a specific algorithmic question

- Only one ship S in an unbounded grid
(no information comes from a position close to the boundary of the grid)

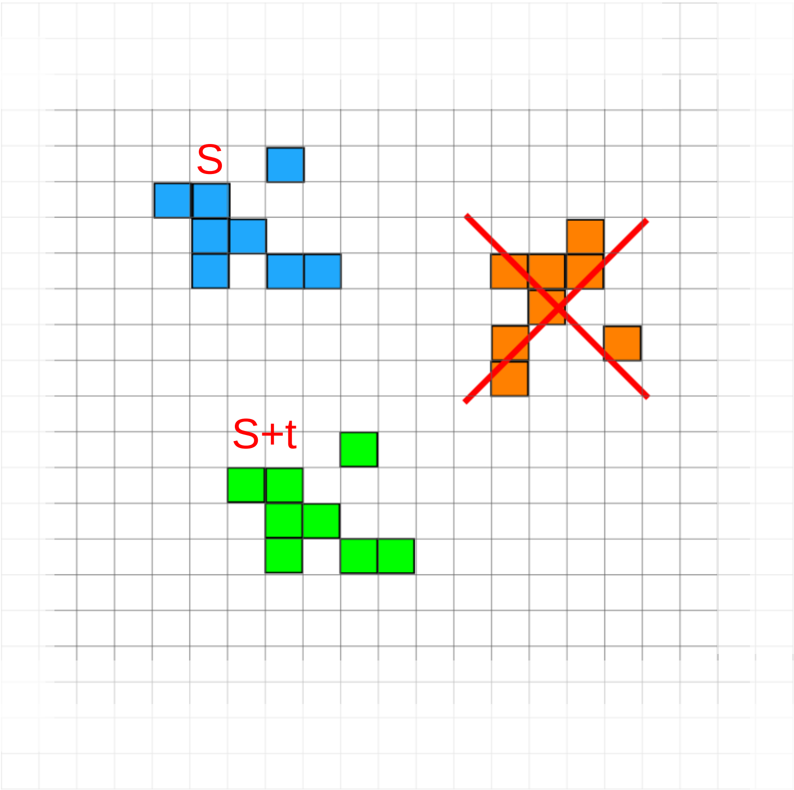
- The shape of the ship is given but there is no restriction on it…



- Only one ship S in an unbounded grid
(no information comes from a position close to the boundary of the grid)

- The shape of the ship is given but there is no restriction on it…

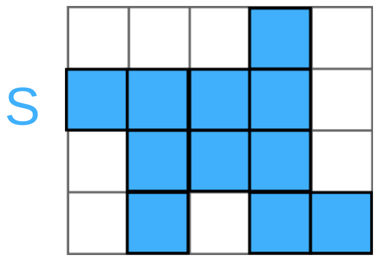
- Moving by translation is allowed but NOT rotations



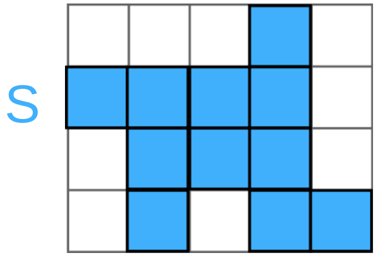
- Only one ship S in an unbounded grid
(no information comes from a position close to the boundary of the grid)

- The shape of the ship is given but there is no restriction on it…

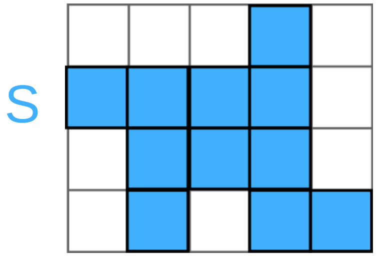
- Moving by translation is allowed but NOT rotations



Shape of the ship

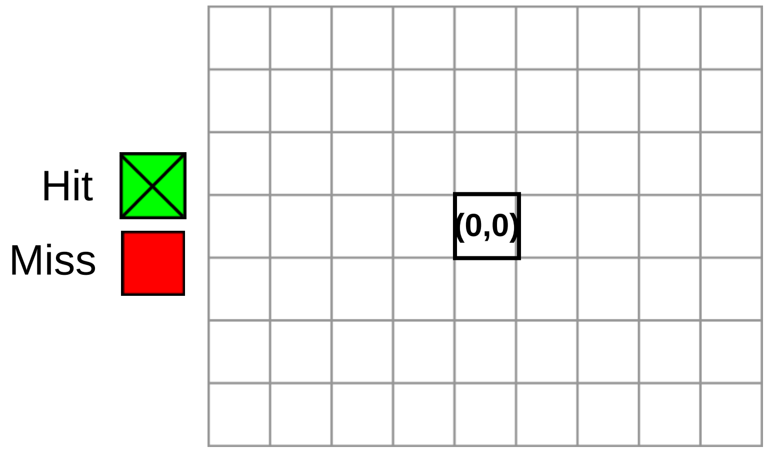


Shape of the ship

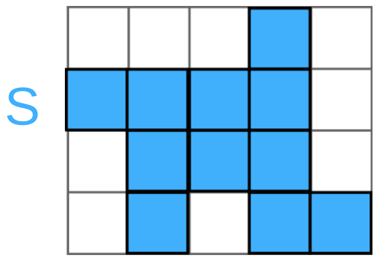


Shape of the ship

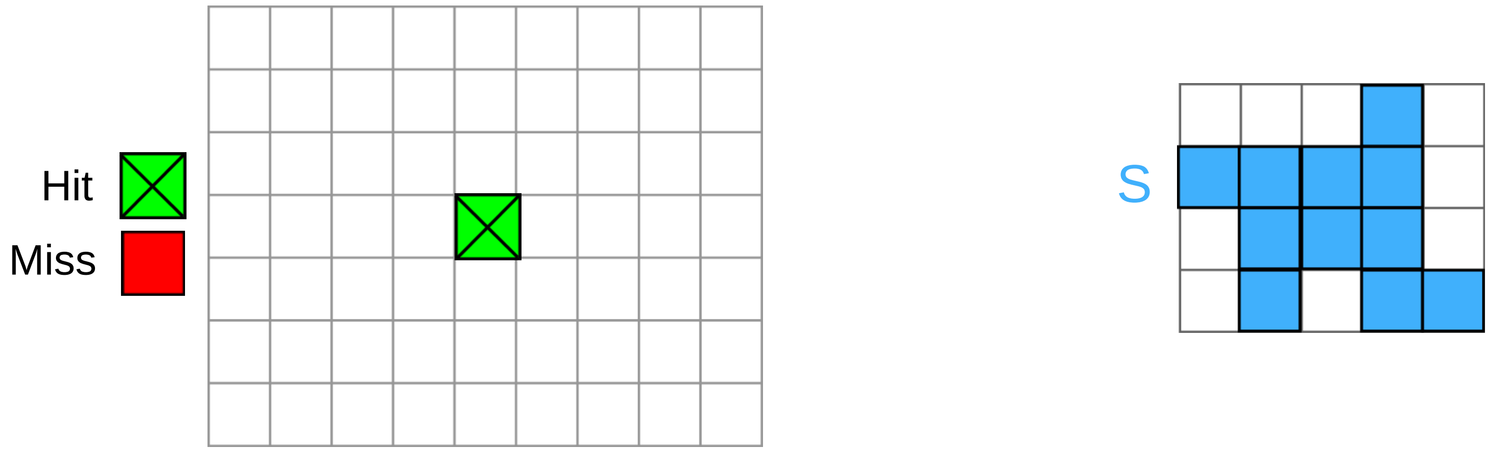
n=|S|=11



Shooting grid

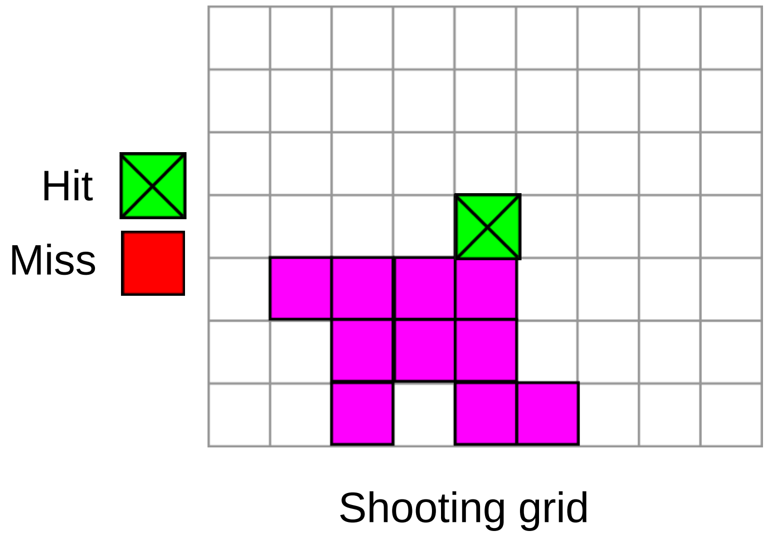


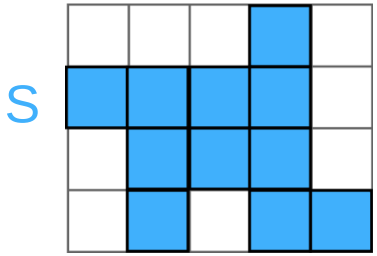
Shape of the ship



Shooting grid

We assume that the ship has been hit one time by shooting at coordinates (0,0)

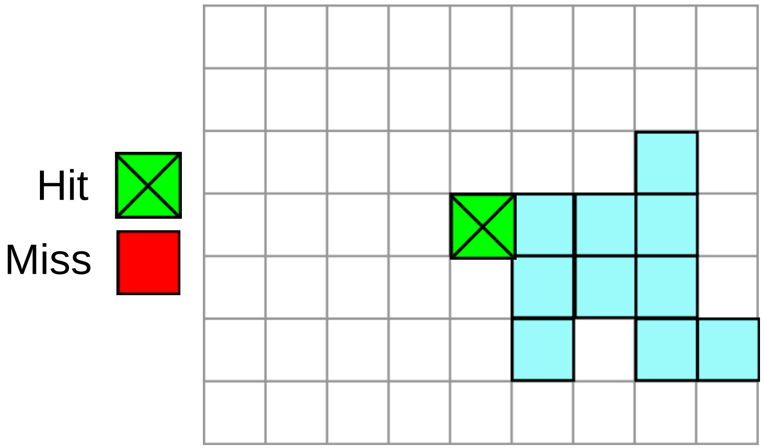




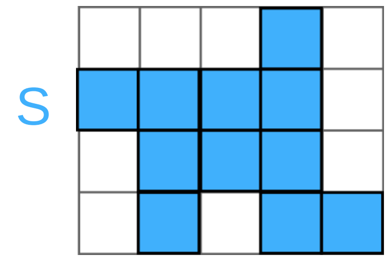
We assume that the ship has been hit one time by shooting at coordinates (0,0)

I Problem statement

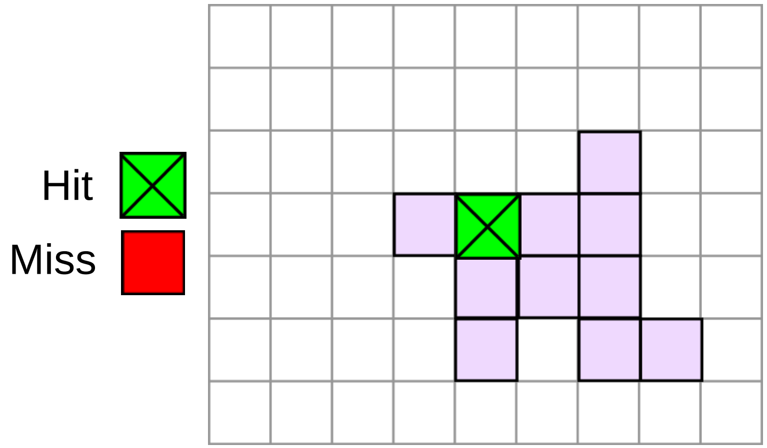
The position of the ship is unknown



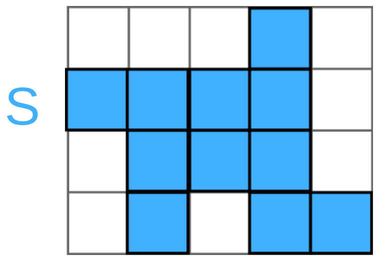
Shooting grid



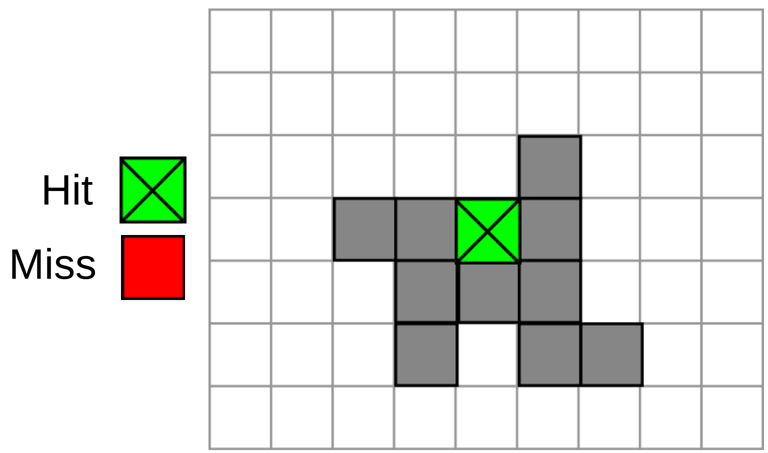
We assume that the ship has been hit one time by shooting at coordinates (0,0)



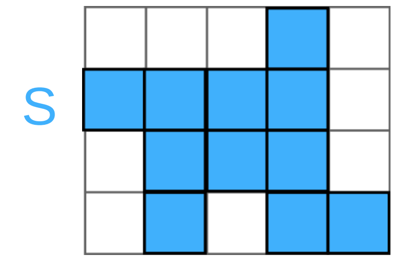
Shooting grid



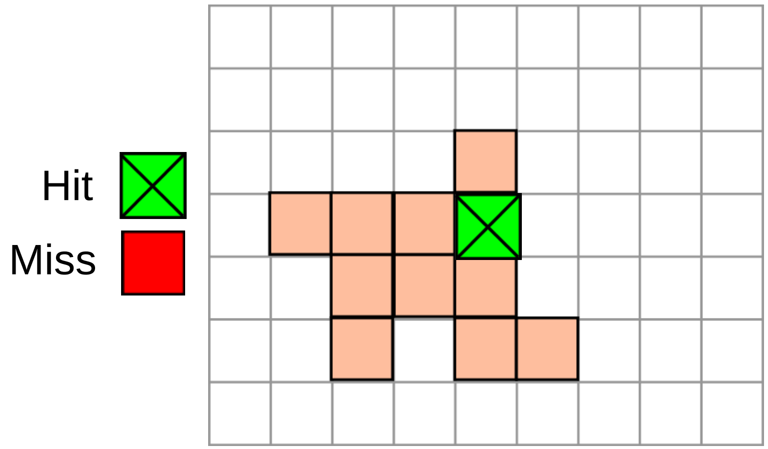
We assume that the ship has been hit one time by shooting at coordinates (0,0)



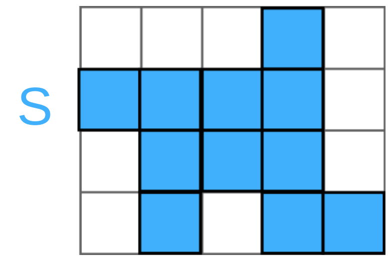
Shooting grid



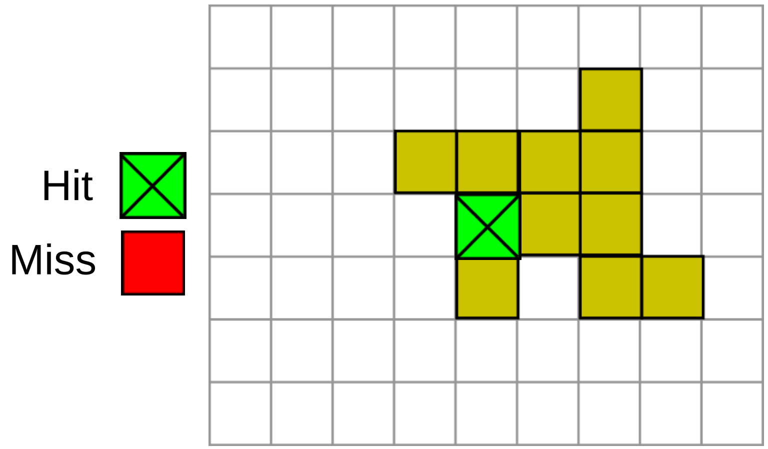
We assume that the ship has been hit one time by shooting at coordinates (0,0)



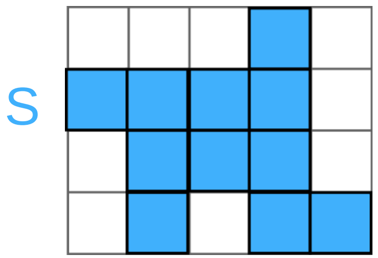
Shooting grid



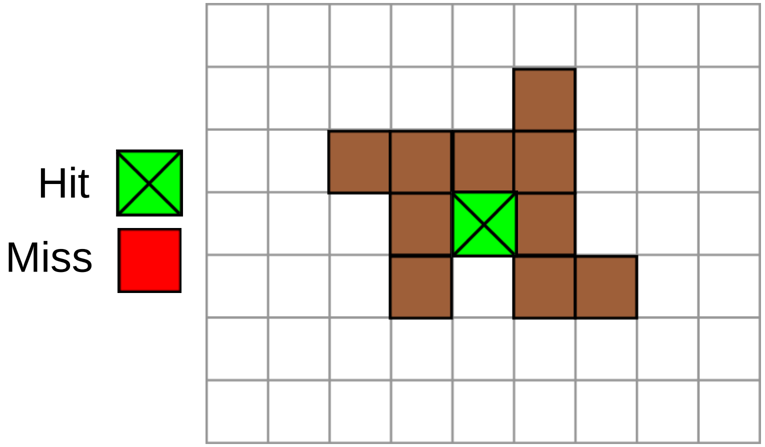
We assume that the ship has been hit one time by shooting at coordinates (0,0)



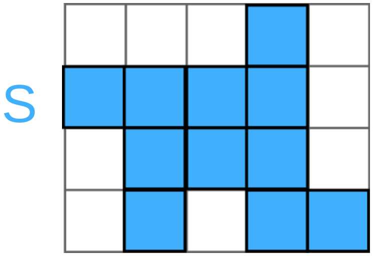
Shooting grid



We assume that the ship has been hit one time by shooting at coordinates (0,0)



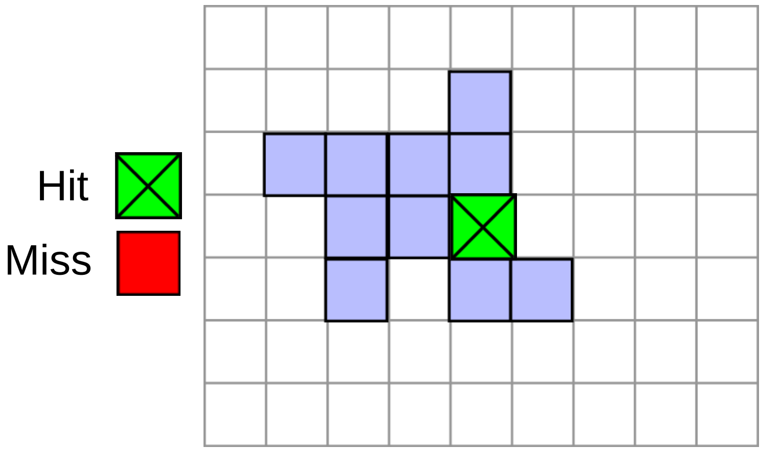
Shooting grid



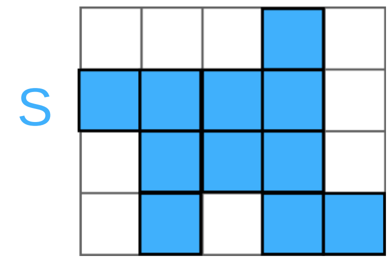
We assume that the ship has been hit one time by shooting at coordinates (0,0)

I Problem statement

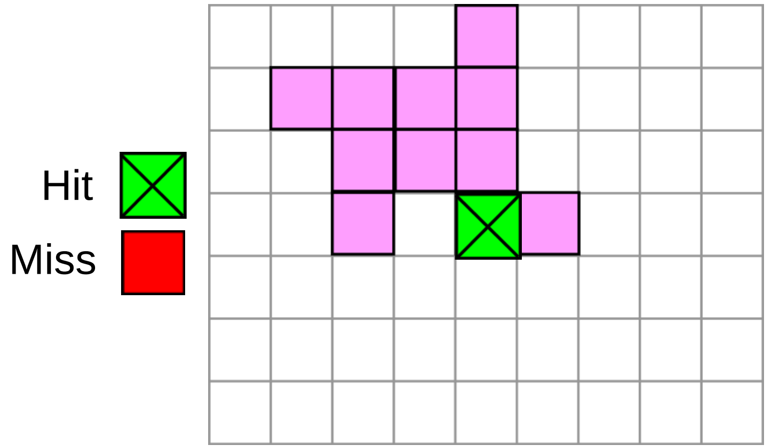
The position of the ship is unknown



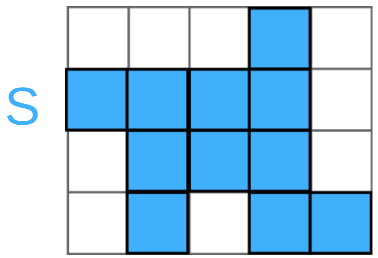
Shooting grid



We assume that the ship has been hit one time by shooting at coordinates (0,0)

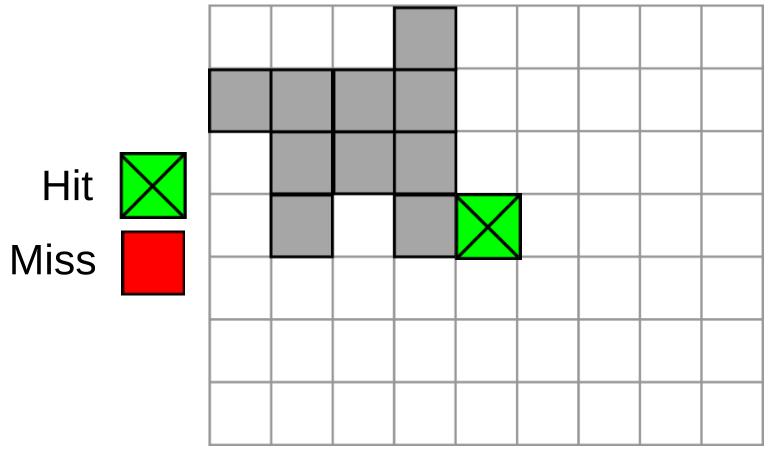


Shooting grid

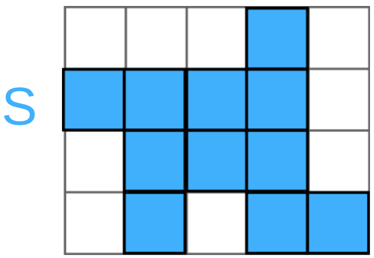


We assume that the ship has been hit one time by shooting at coordinates (0,0)

The position of the ship is unknown

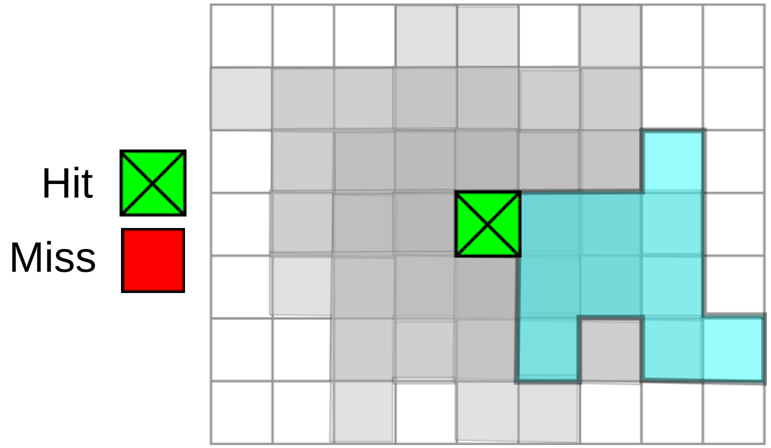


Shooting grid



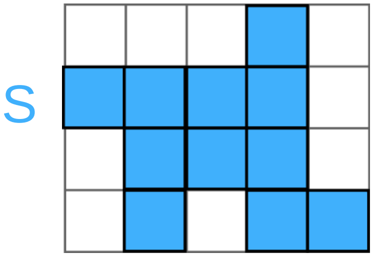
We assume that the ship has been hit one time by shooting at coordinates (0,0)

The position of the ship is unknown

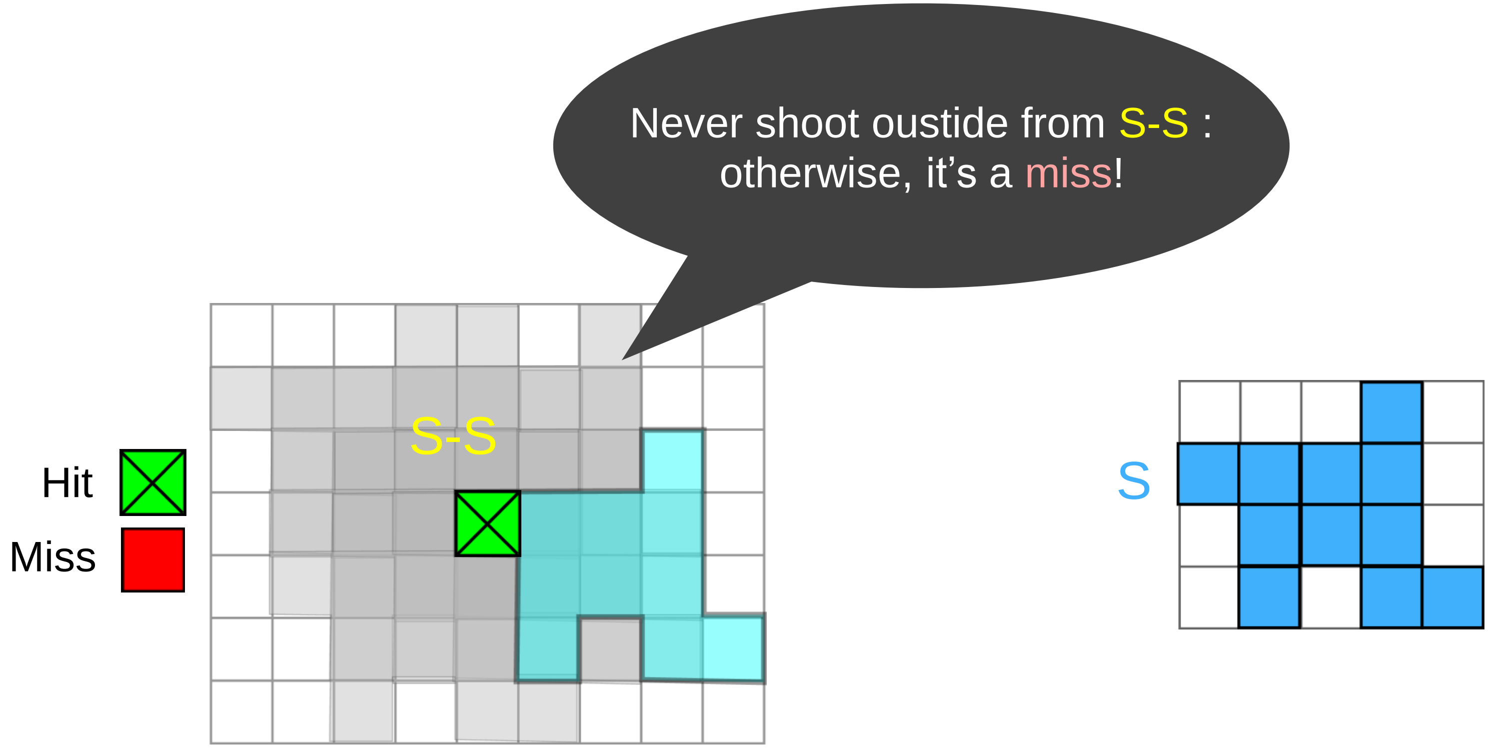


Shooting grid

n positions are possible…



We assume that the ship has been hit one time by shooting at coordinates (0,0)

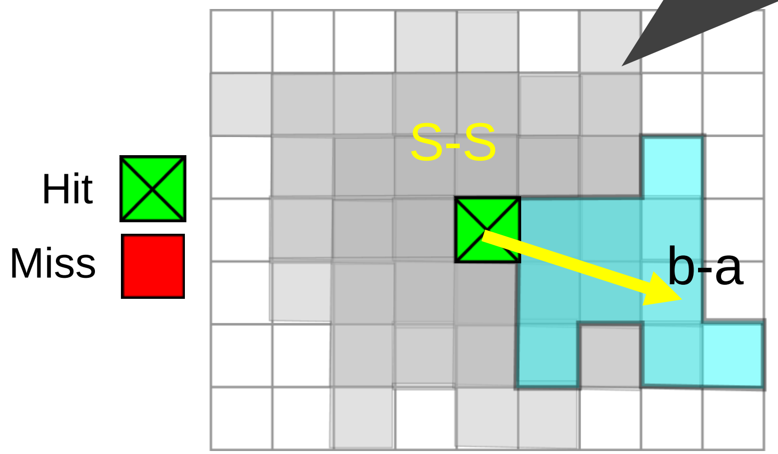


Shooting grid

n positions are possible…

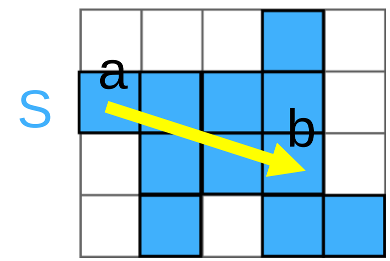
We assume that the ship has been hit one time by shooting at coordinates (0,0)

Never shoot oustide from S-S :
otherwise, it's a miss!

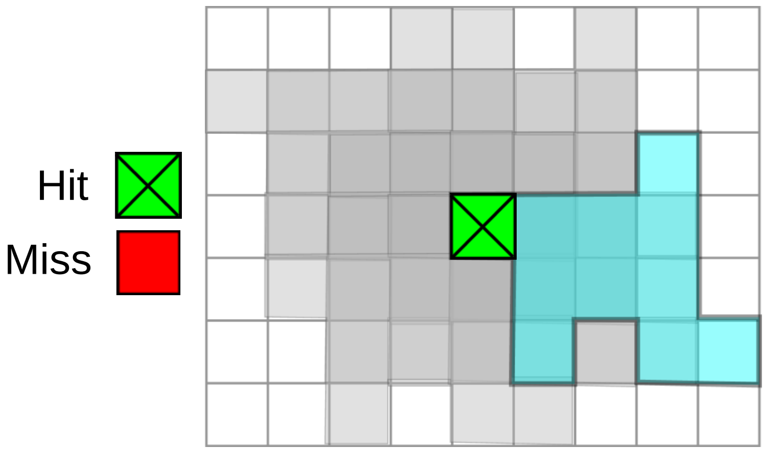


Shooting grid

n positions are possible…

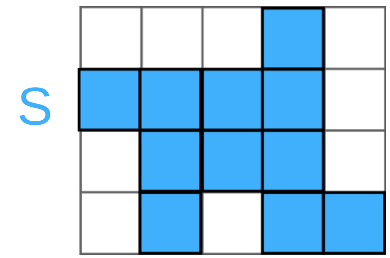


We assume that the ship has been hit one time by shooting at coordinates (0,0)



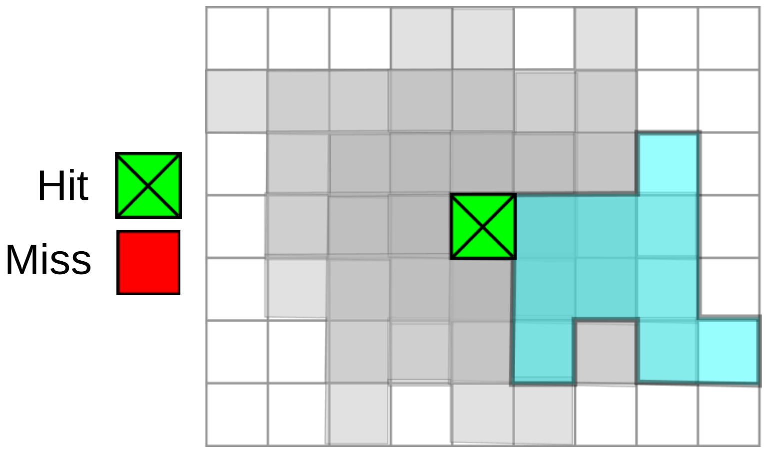
Shooting grid

n positions are possible…



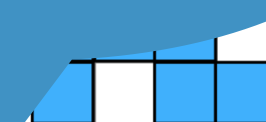
Now, the goal is :
sink the ship with the fewer number of shots as possible

We assume that the ship has been hit one time by shooting at coordinates (0,0)



Shooting grid

n positions are possible…



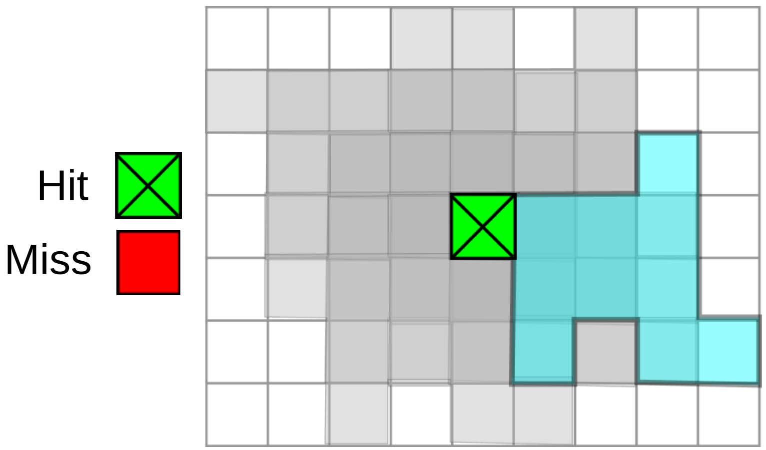
translation:
sink the ship with the fewer number of misses as possible

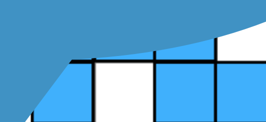
Now, the goal is :
sink the ship with the fewer number of shots as possible

We assume that the ship has been hit one time by shooting at coordinates (0,0)

Find the position of the ship with the fewer number of misses as possible

translation:
sink the ship with the fewer number of misses as possible





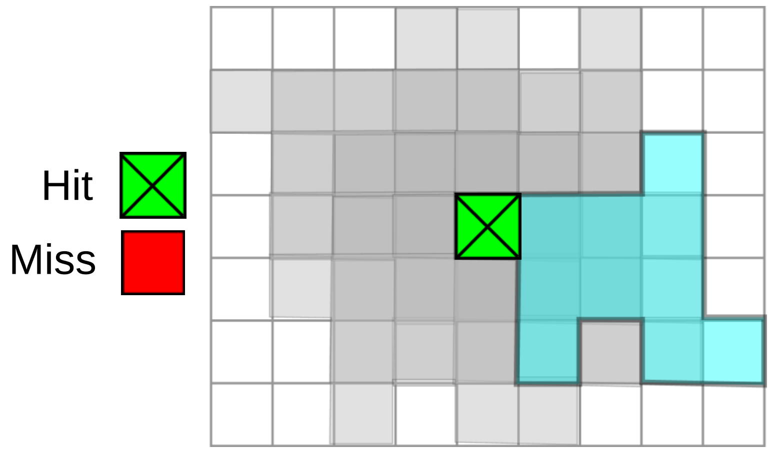
Shooting grid

n positions are possible…

Now, the goal is :
sink the ship with the fewer number of shots as possible

We assume that the ship has been hit one time by shooting at coordinates (0,0)

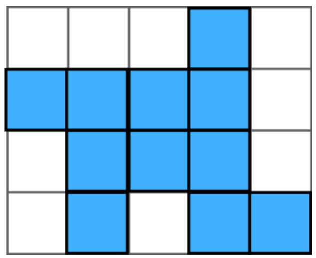
Find the position of the ship with the fewer number of misses as possible



Shooting grid

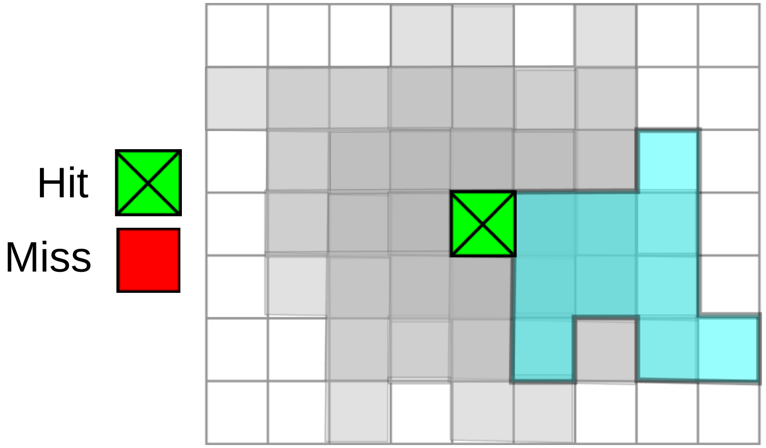
n positions are possible…

S

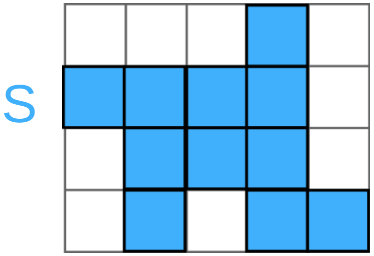


We assume that the ship has been hit one time by shooting at coordinates (0,0)

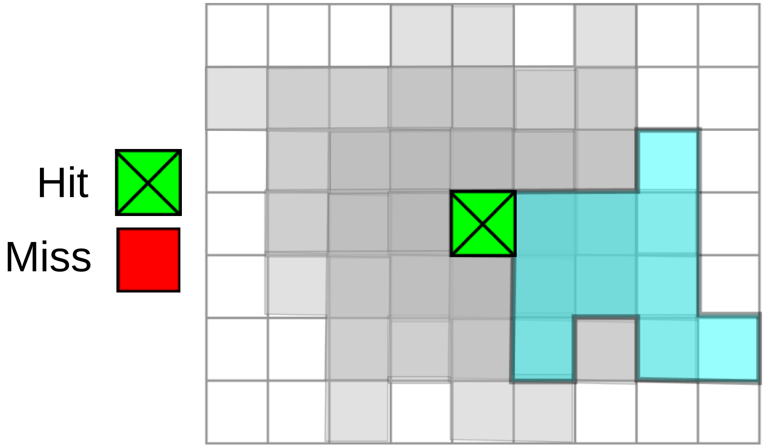
Find the position of the ship with the fewer number of misses as possible



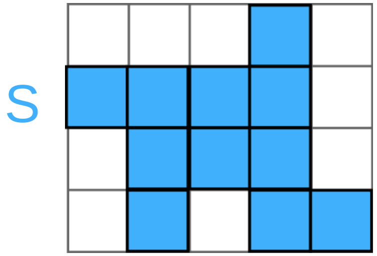
Shooting grid



Find the position of the ship with the fewer number of misses as possible



Shooting grid



Design a shooting algorithm to find the position of the ship with the fewer number of misses…

Plan

I Problem Statement

II Shooting Algorithms

III Results

IV Proofs

Plan

I Problem Statement

II Shooting Algorithms

III Results

IV Proofs

Given a shape, our goal is to design a shooting algorithm
to find the position of the ship
with the fewer number of misses…

Given a shape, our goal is to design a shooting algorithm
to find the position of the ship
with the fewer number of misses…

2 different ways to
model an algorithm

With shooting rules
(for simple algorithms)

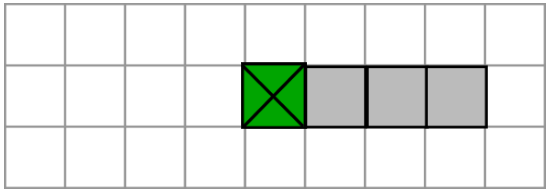
With a decision tree
(when there are many cases)

Shape of the ship

n=|S|=4

Shape of the ship

n=|S|=4

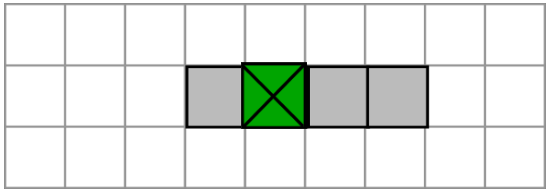


Shooting grid

4 positions are possible…

Shape of the ship

n=|S|=4

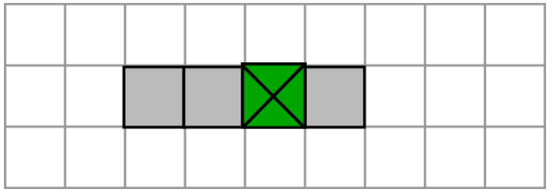


Shooting grid

4 positions are possible…

Shape of the ship

n=|S|=4

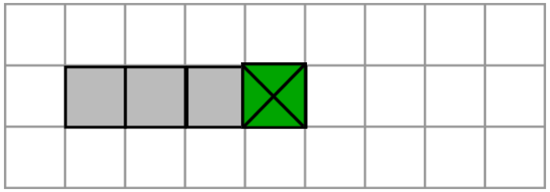


Shooting grid

4 positions are possible…

Shape of the ship

n=|S|=4

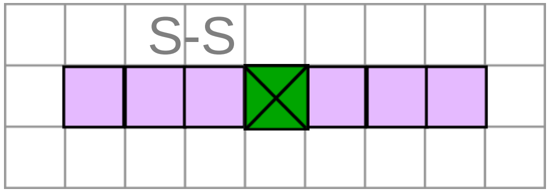


Shooting grid

4 positions are possible…

Shape of the ship

n=|S|=4



Shooting grid

4 positions are possible…

Shoot (1,0) ?

Shoot (2,0) ?

Shoot (3,0) ?

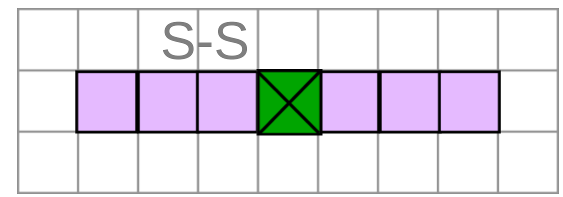
Shoot (-1,0) ?

Shoot (-2,0) ?

Shoot (-3,0) ?

Shape of the ship

n=|S|=4



Shooting grid

4 positions are possible…

Choose a new shot

Shoot (1,0) ?

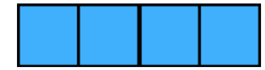
Shoot (2,0) ?

Shoot (3,0) ?

Shoot (-1,0) ?

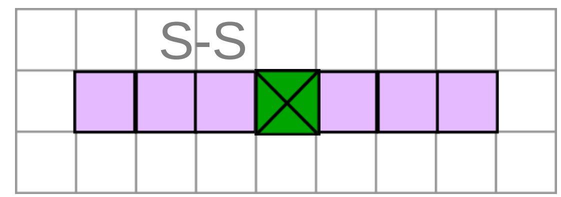
Shoot (-2,0) ?

Shoot (-3,0) ?



Shape of the ship

n=|S|=4



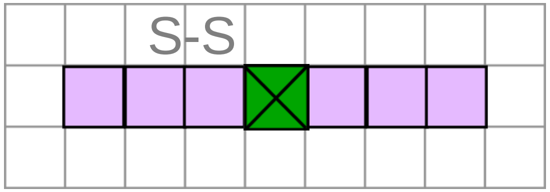
Shooting grid

4 positions are possible…

What is the best strategy ?

Shape of the ship

n=|S|=4



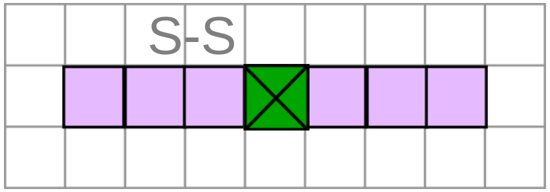
Shooting grid

4 positions are possible…

What is the best strategy ?

Shape of the ship

n=|S|=4



Shooting grid

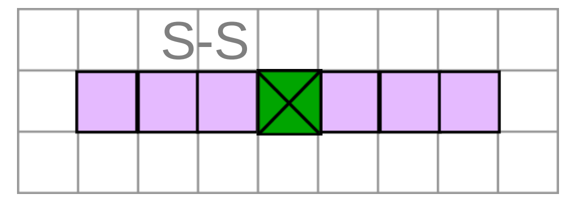
4 positions are possible…

A shooting algorithm:
Shoot on the left of the previous shot until having a miss.
Then the position of the ship is known.

What is the best strategy ?

Shape of the ship

n=|S|=4



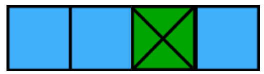
Shooting grid

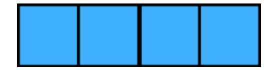
4 positions are possible…

A shooting algorithm:

Shoot on the left of the previous shot until having a miss.
Then the position of the ship is known.

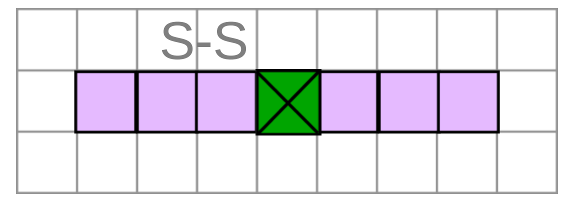
What is the best strategy ?





Shape of the ship

n=|S|=4

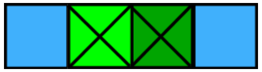


Shooting grid

4 positions are possible…

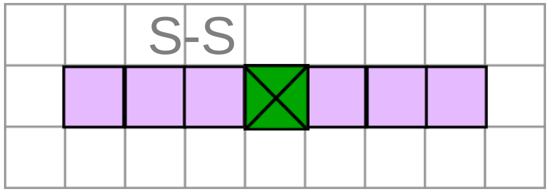
A shooting algorithm:
Shoot on the left of the previous shot until having a miss.
Then the position of the ship is known.

What is the best strategy ?



Shape of the ship

n=|S|=4

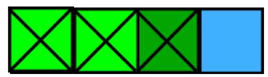


A shooting algorithm:
Shoot on the left of the previous shot until having a miss.
Then the position of the ship is known.

Shooting grid

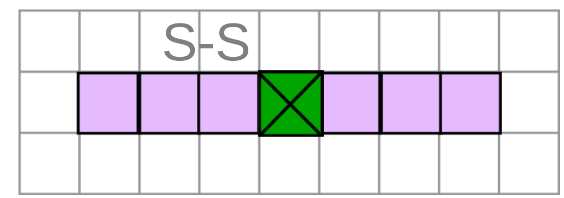
4 positions are possible…

What is the best strategy ?



Shape of the ship

n=|S|=4

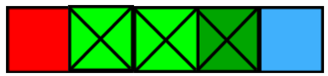


A shooting algorithm:
Shoot on the left of the previous shot until having a miss.
Then the position of the ship is known.

Shooting grid

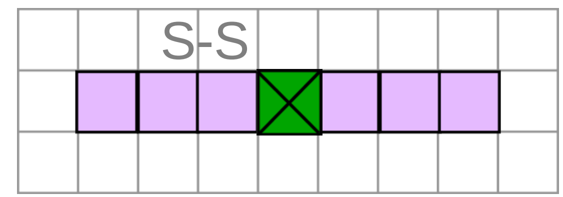
4 positions are possible…

What is the best strategy ?



Shape of the ship

n=|S|=4



Shooting grid

4 positions are possible…

A shooting algorithm:

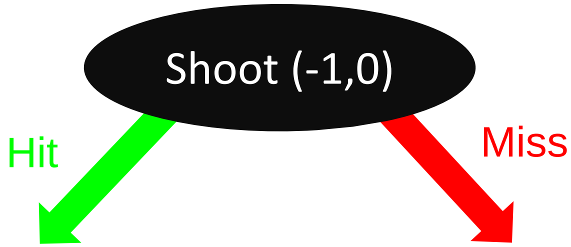
Shoot on the left of the previous shot until having a miss.

Then the position of the ship is known.

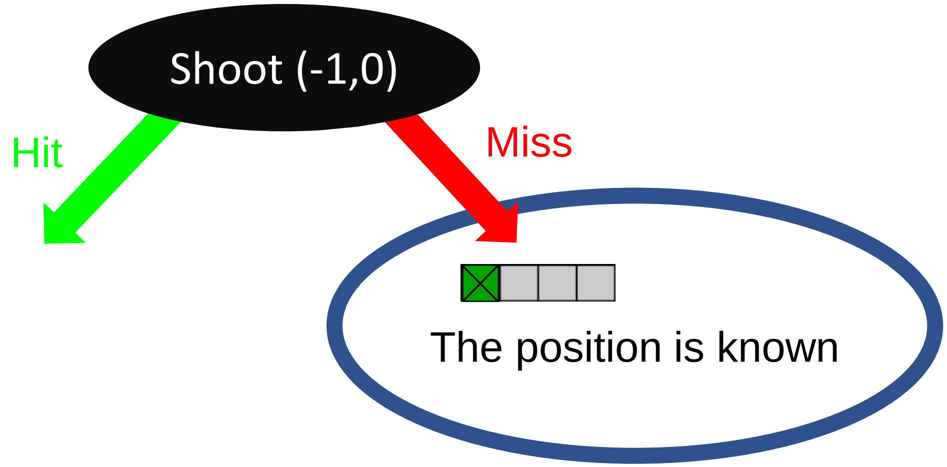
What is the best strategy ?

Shoot (-1,0)

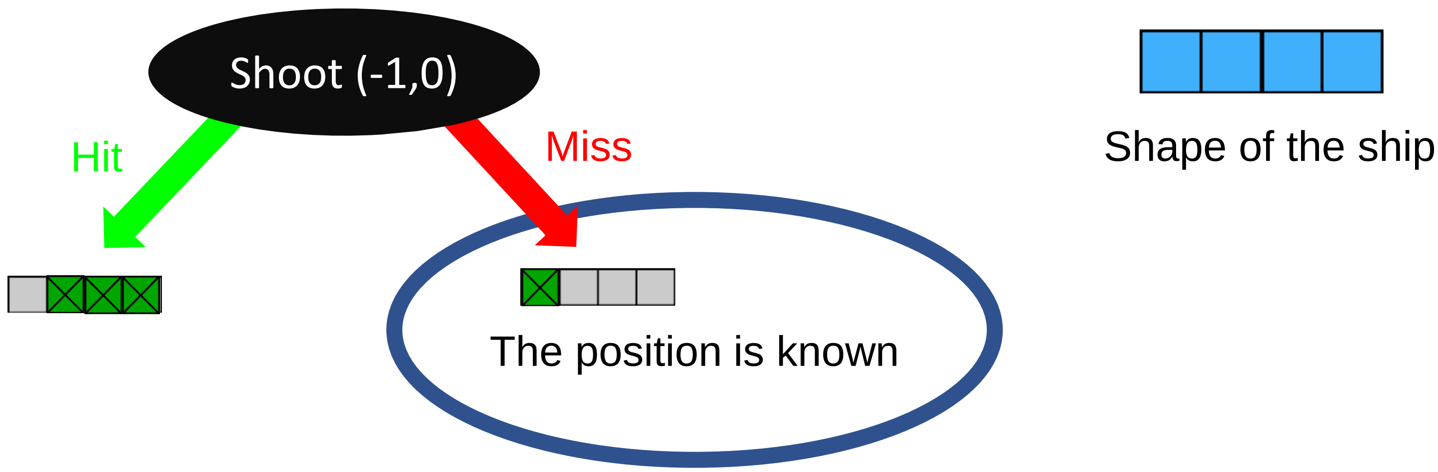
Shape of the ship

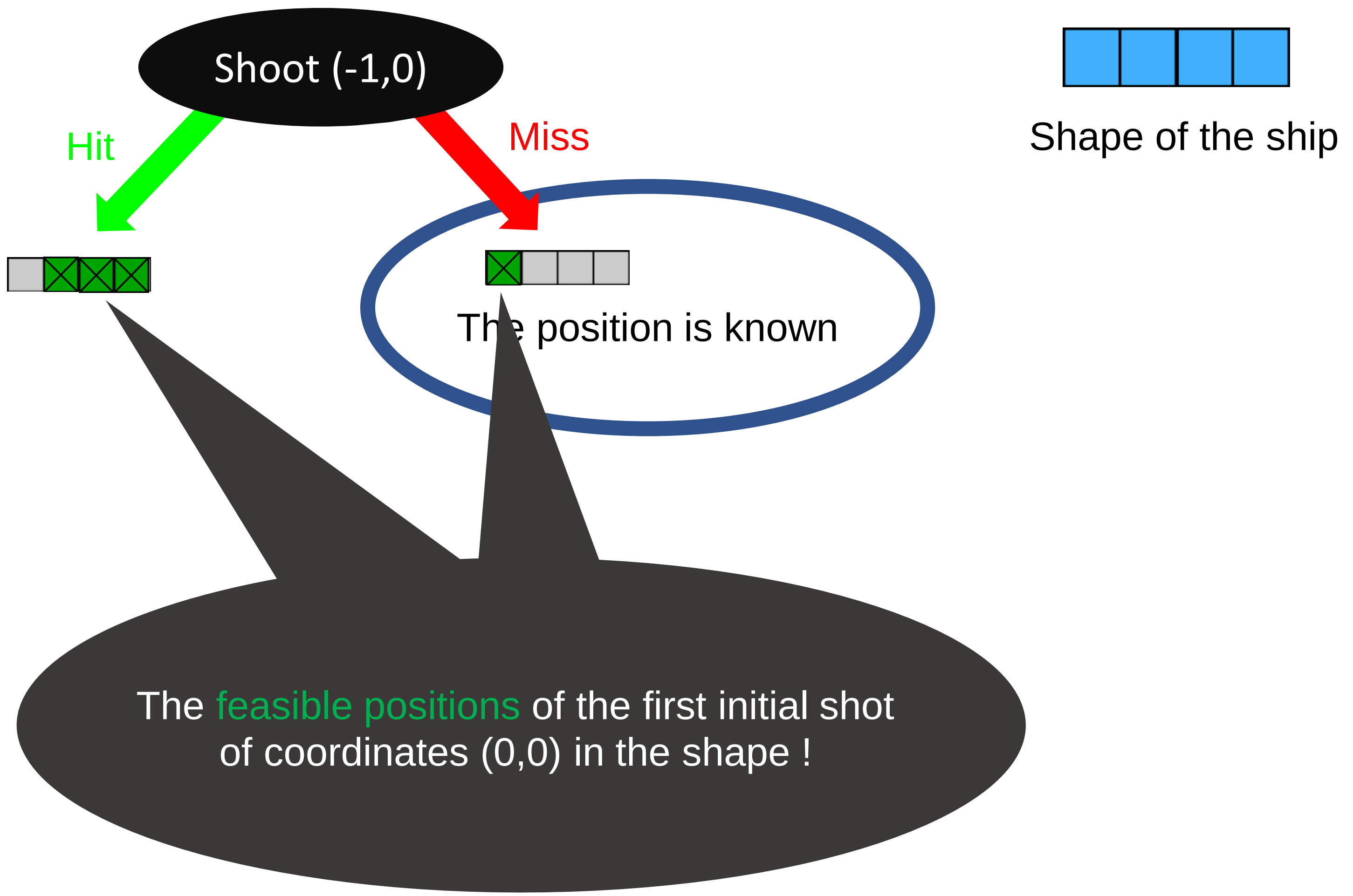


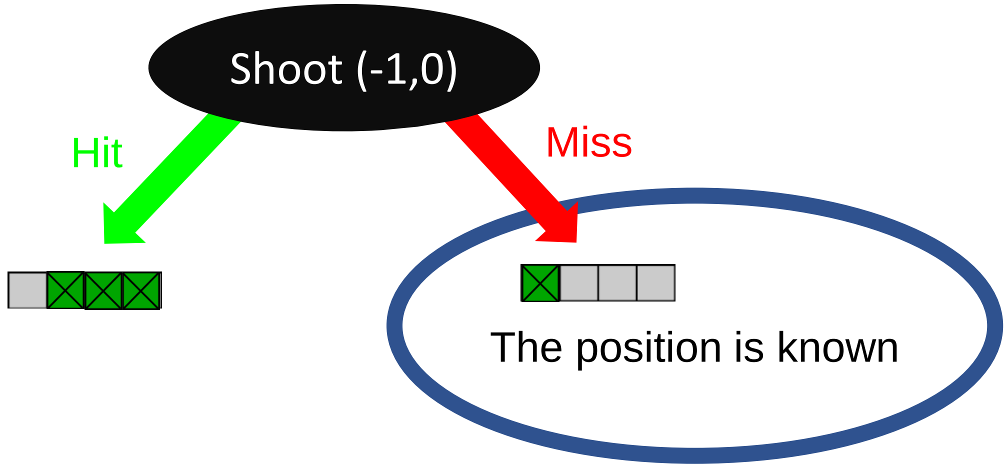
Shape of the ship



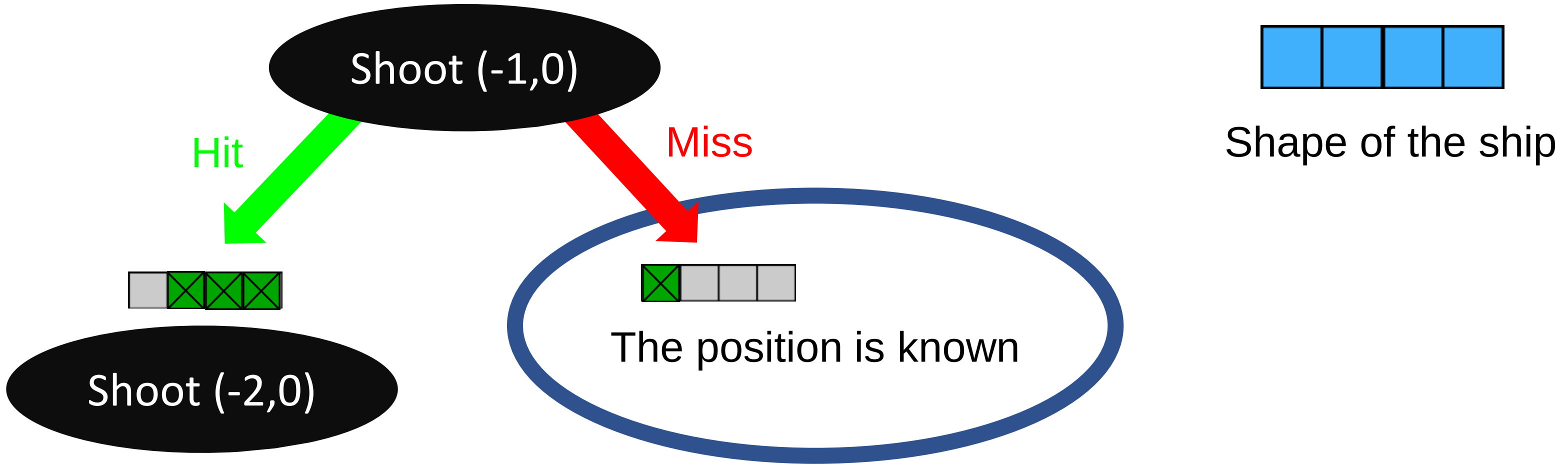
Shape of the ship

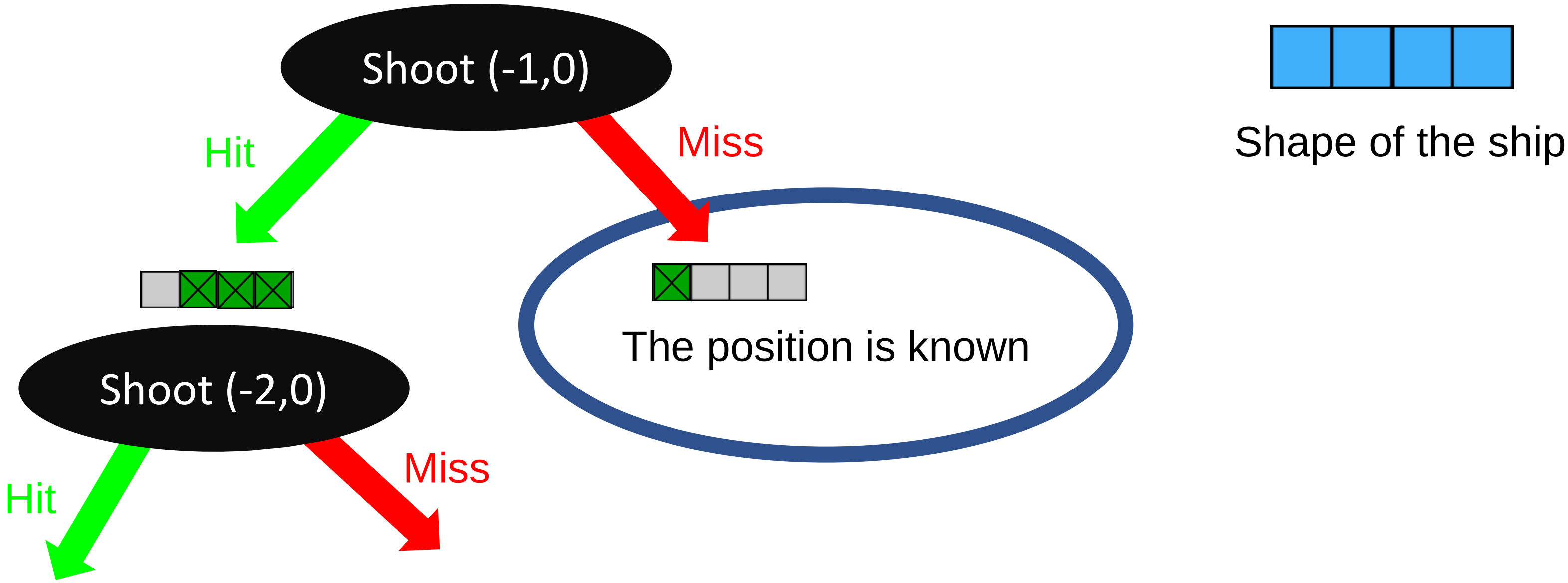

Shoot (-1,0)
Hit
Miss
The position is known
Shape of the ship

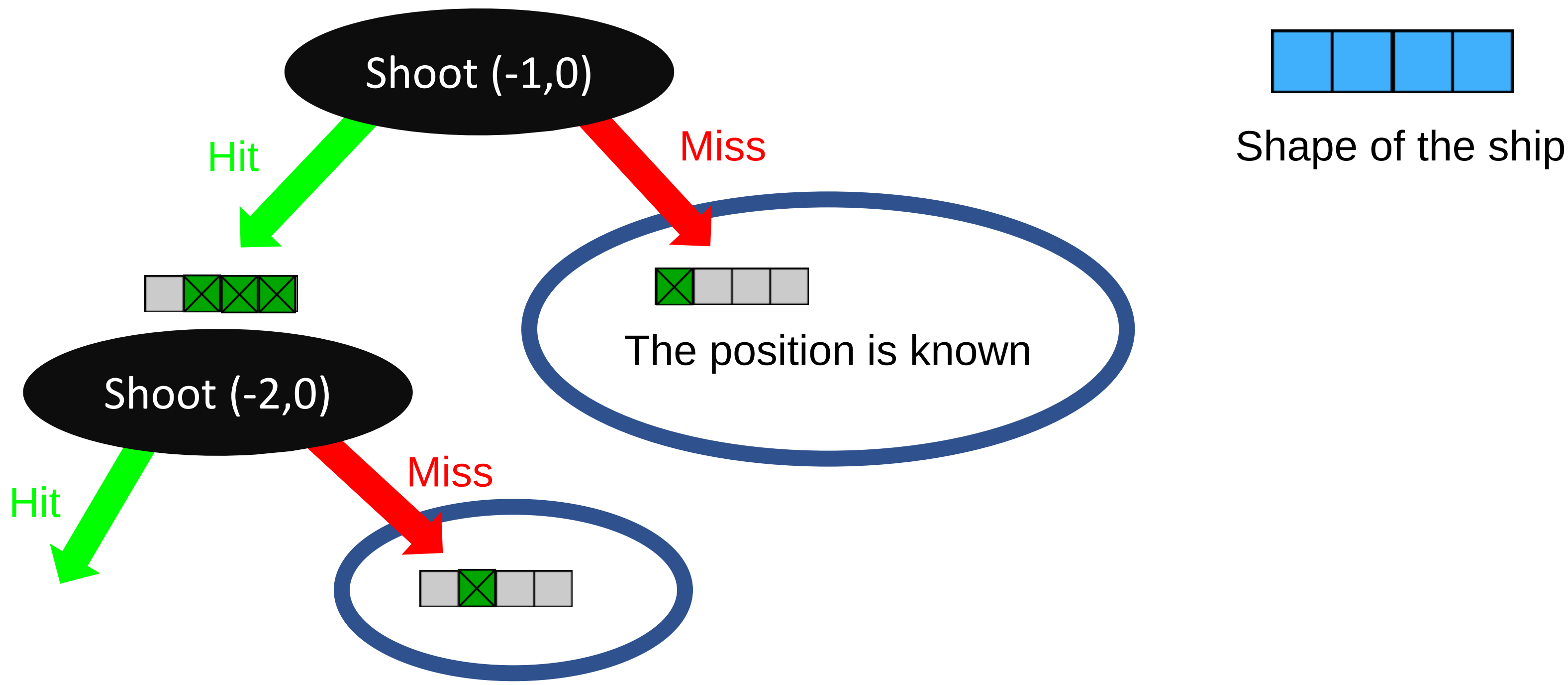

Shoot (-1,0)
Hit
Miss
Shape of the ship
The position is known
The feasible positions of the first initial shot
of coordinates (0,0) in the shape !

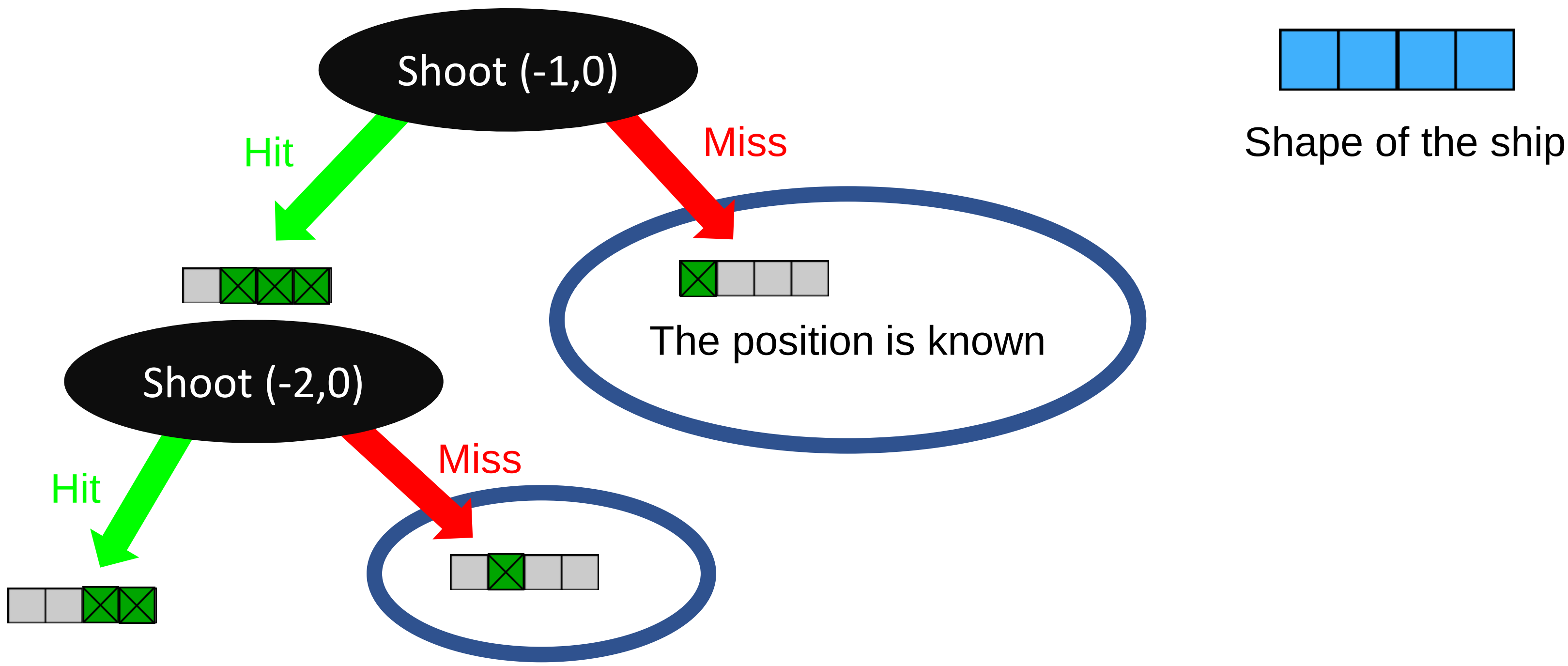

Shoot (-1,0)
Hit
Miss
The position is known

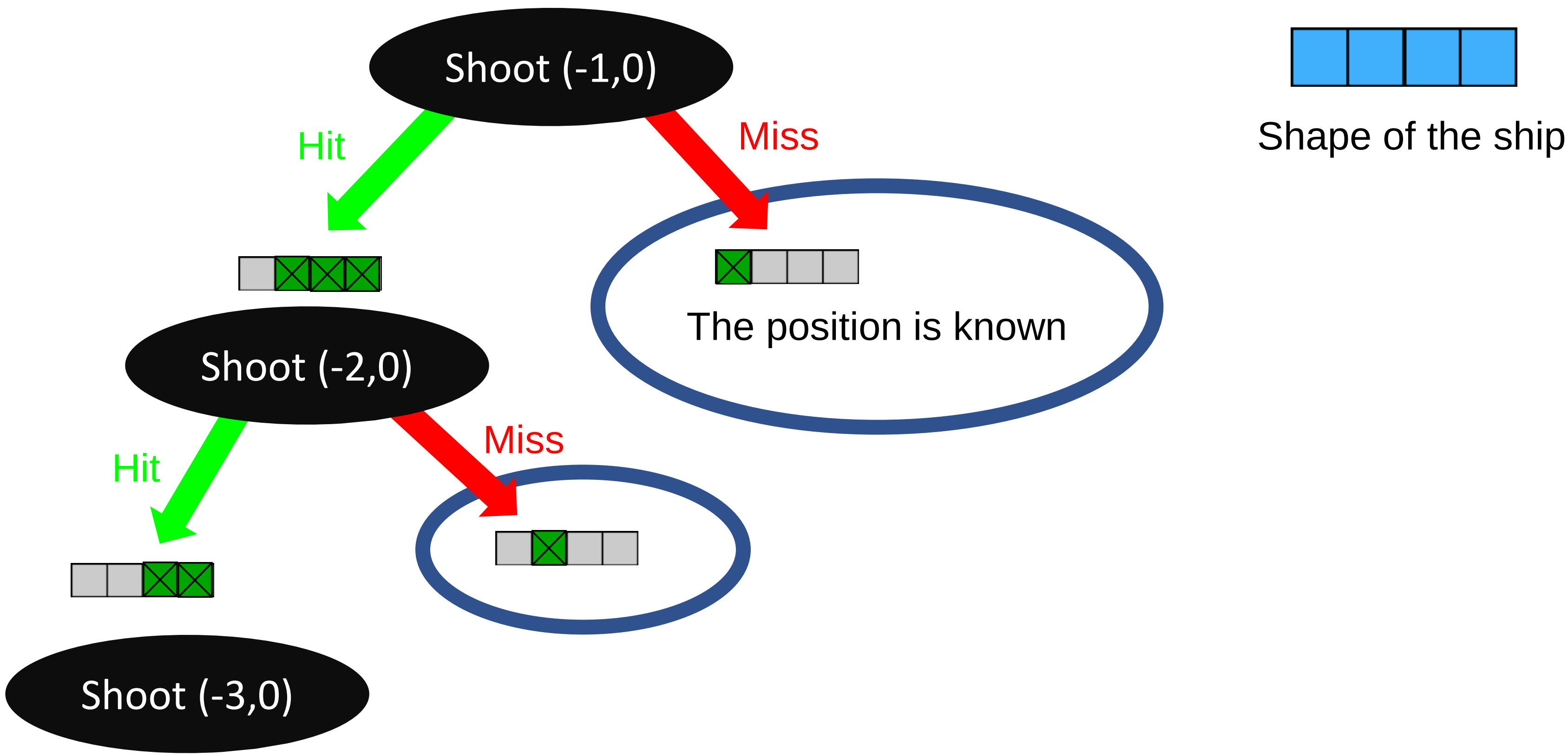

Shape of the ship

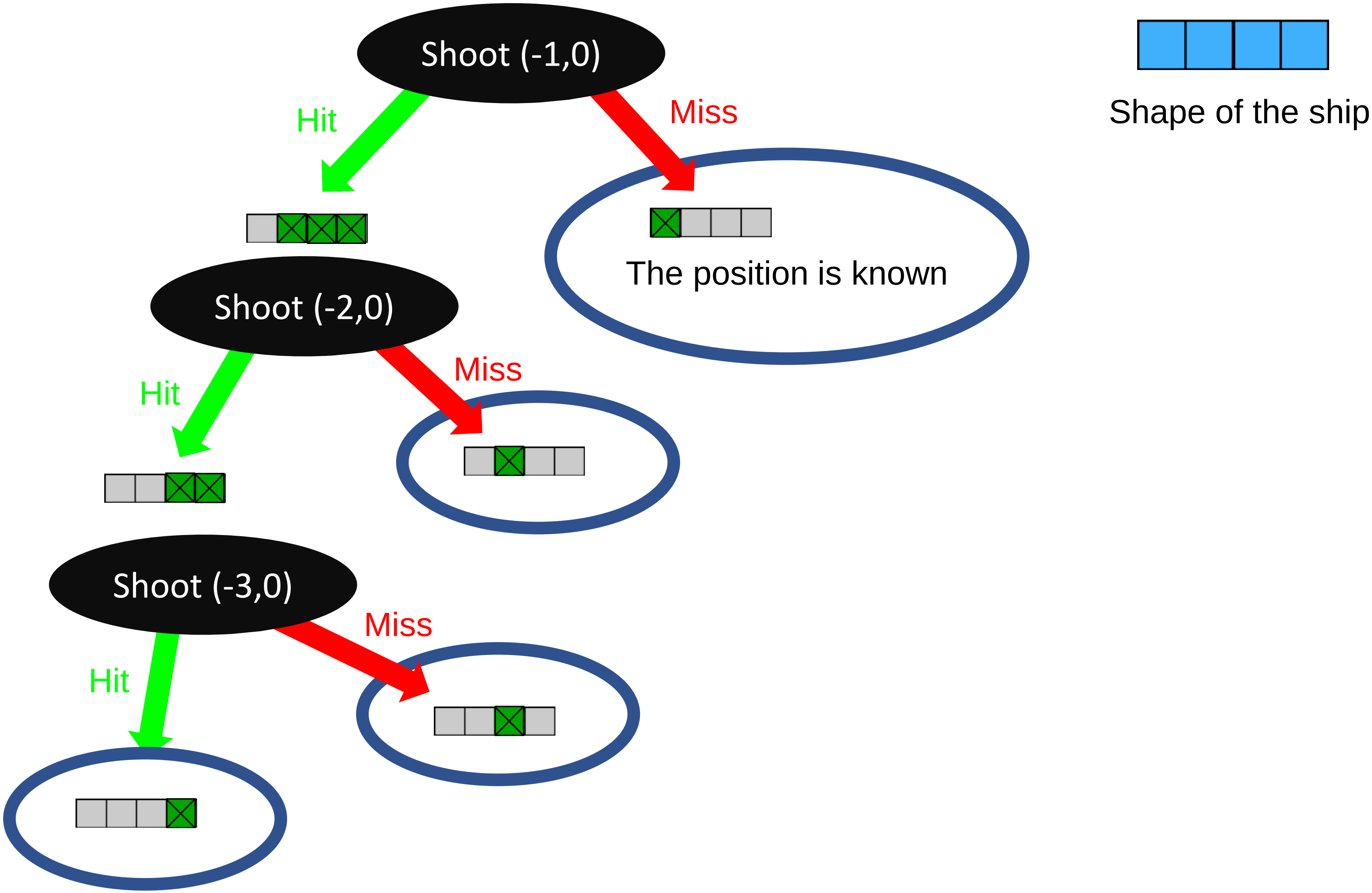

Shoot (-1,0)
Hit
Miss
Shoot (-2,0)
The position is known
Shape of the ship

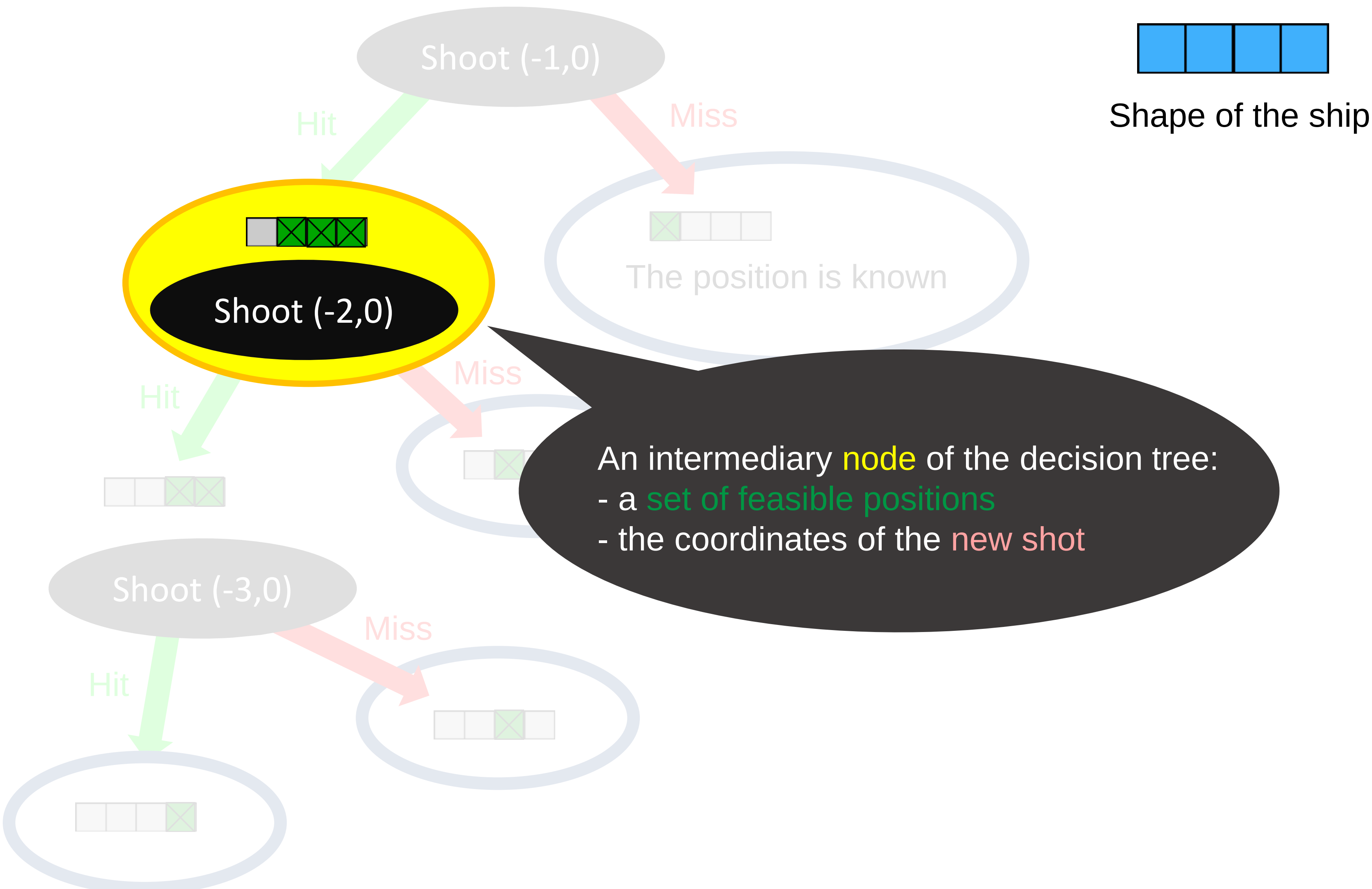

Shoot (-1,0)
Hit
Miss
Shoot (-2,0)
Hit
Miss
The position is known
Shape of the ship

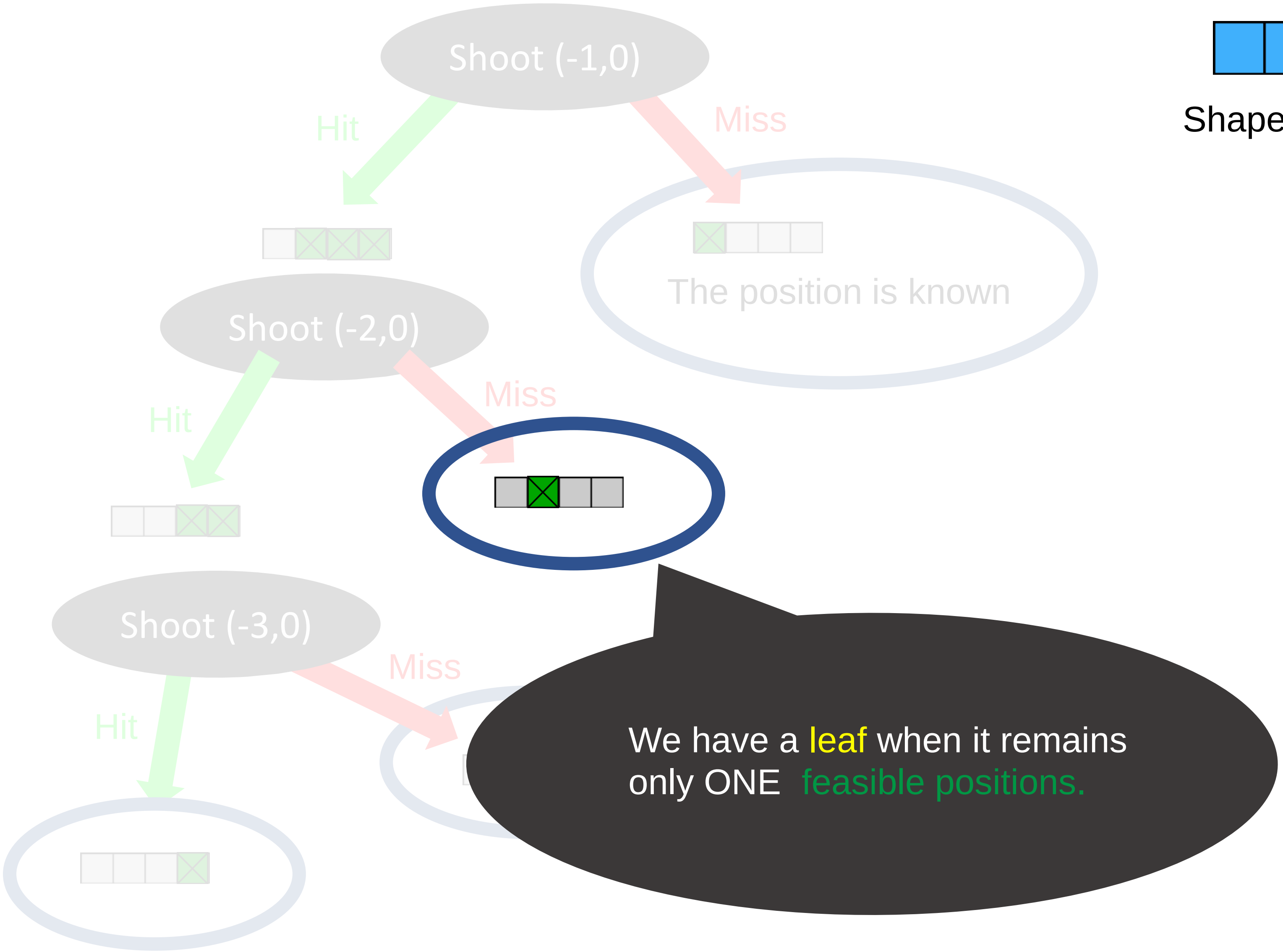

Shoot (-1,0)
Hit
Miss
The position is known
Shoot (-2,0)
Hit
Miss
Shape of the ship

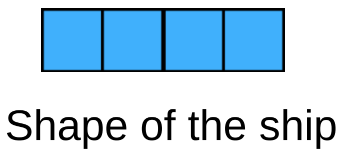

Shoot (-1,0)
Hit
Miss
Shape of the ship
The position is known
Shoot (-2,0)
Hit
Miss


Shoot (-1,0)
Hit
Miss
Shoot (-2,0)
The position is known
Hit
Miss
Shoot (-3,0)
Shape of the ship


Shoot (-1,0)
Hit
Miss
The position is known
Shoot (-2,0)
Hit
Miss
Shoot (-3,0)
Hit
Miss
Shape of the ship


Shoot (-1,0)
Hit
Miss
The position is known
Shoot (-2,0)
Hit
Miss
Shoot (-3,0)
Hit
Miss
Shape of the ship
An intermediary node of the decision tree:
- a set of feasible positions
- the coordinates of the new shot

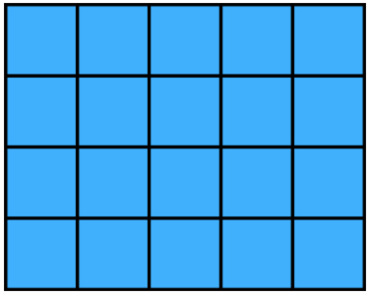

Shoot (-1,0)
Hit
Miss
The position is known
Shoot (-2,0)
Hit
Miss
Shoot (-3,0)
Hit
Miss
We have a leaf when it remains only ONE feasible positions.


Shape of the ship

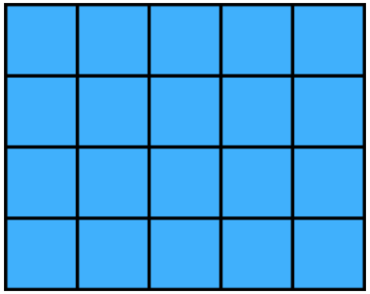
2 different ways to
model an algorithm

With shooting rules
(for simple algorithms)

With a decision tree
(when there are many cases)

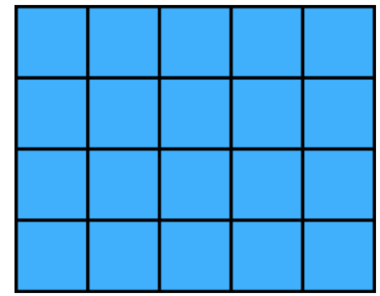


Shape of the ship

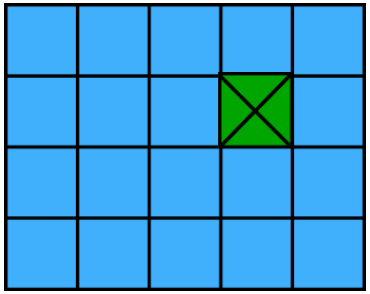


Shape of the ship

What is the best strategy ?

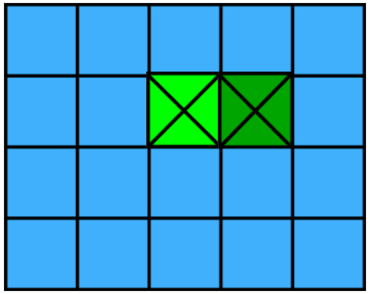


1) Shoot on the left of the previous shot until having a miss.

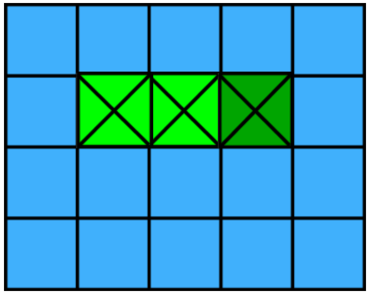


1) Shoot on the left of the previous shot
until having a miss.

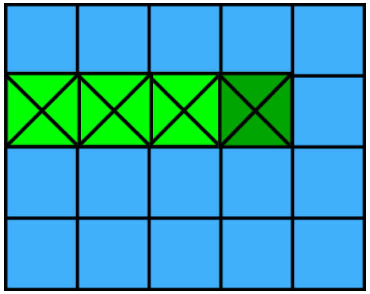
1) Shoot on the left of the previous shot until having a miss.



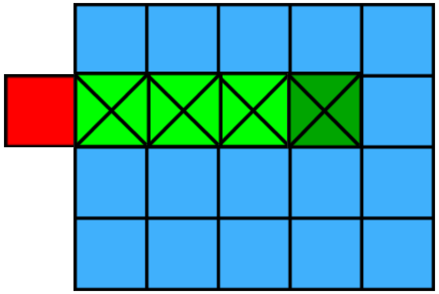
1) Shoot on the left of the previous shot until having a miss.

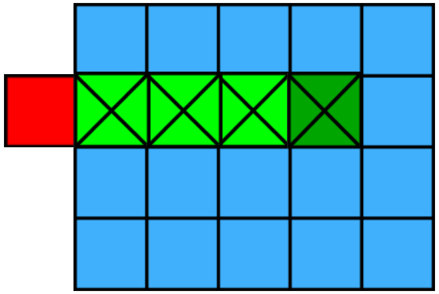


1) Shoot on the left of the previous shot until having a miss.



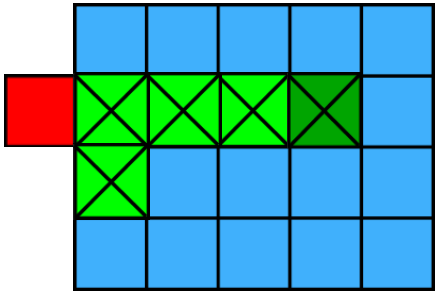
1) Shoot on the left of the previous shot until having a miss.





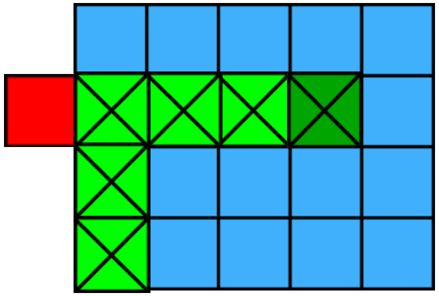
1) Shoot on the left of the previous shot until having a miss.

2) Then shoot down until having a miss.
Then the position of the ship is known.

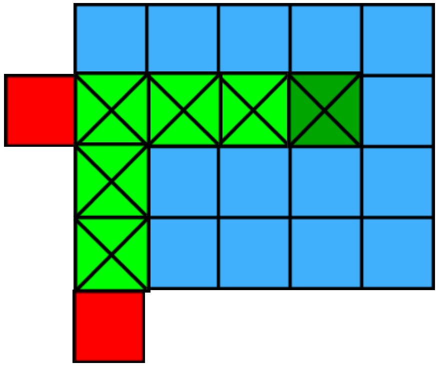


1) Shoot on the left of the previous shot until having a miss.

2) Then shoot down until having a miss. Then the position of the ship is known.



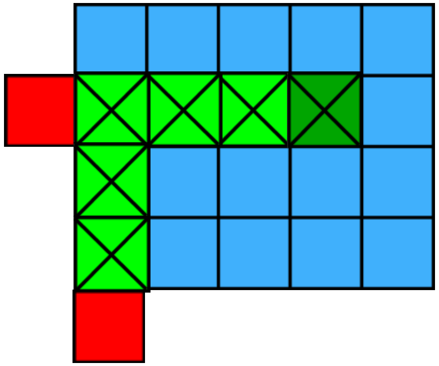
1) Shoot on the left of the previous shot until having a miss.
2) Then shoot down until having a miss.
Then the position of the ship is known.



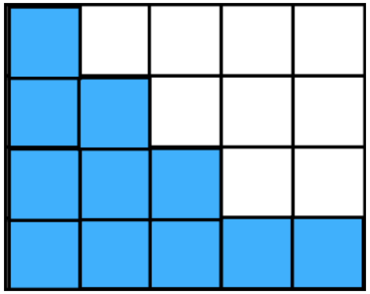
1) Shoot on the left of the previous shot until having a miss.

2) Then shoot down until having a miss. Then the position of the ship is known.

1) Shoot on the left of the previous shot until having a miss.
2) Then shoot down until having a miss.
Then the position of the ship is known.

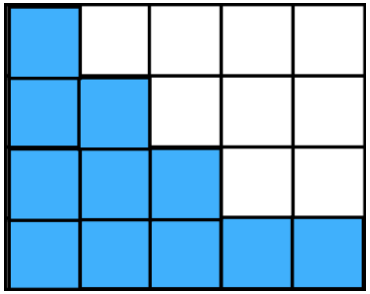


It allows to determine the position of the ship with at most two misses



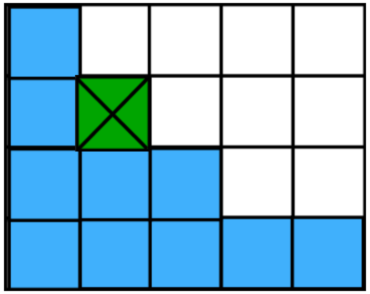
A new shape

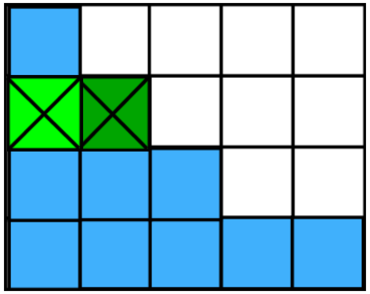
The previous algorithm works: it determines the position with at most 2 misses.



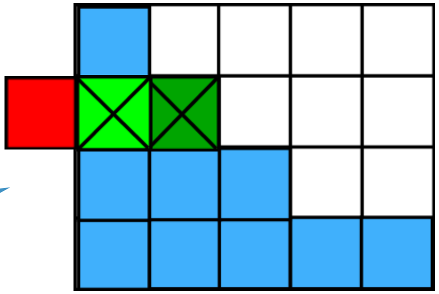
A new shape

The previous algorithm works: it determines the position with at most 2 misses.

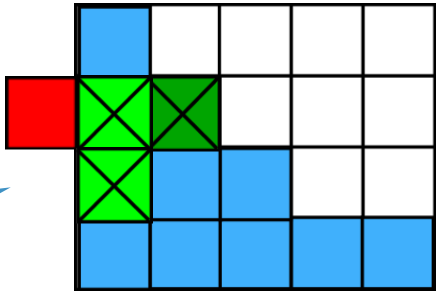




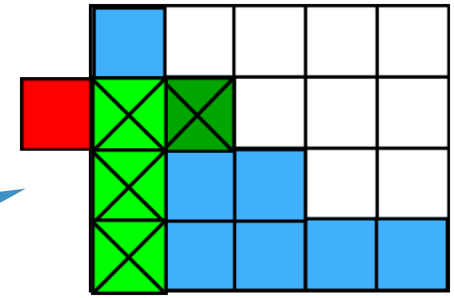
The previous algorithm works: it determines the position with at most 2 misses.



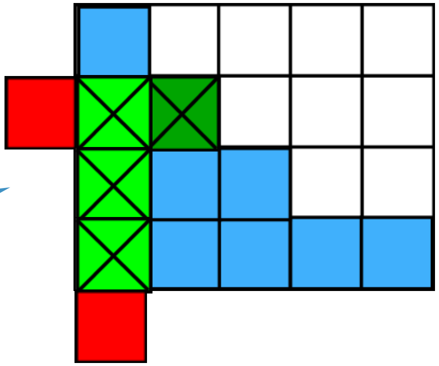
The previous algorithm works: it determines the position with at most 2 misses.



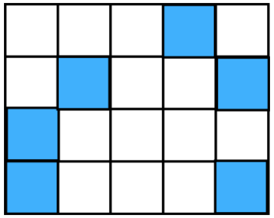
The previous algorithm works: it determines the position with at most 2 misses.



The previous algorithm works: it determines the position with at most 2 misses.

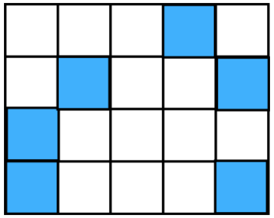


The previous algorithm works: it determines the position with at most 2 misses.

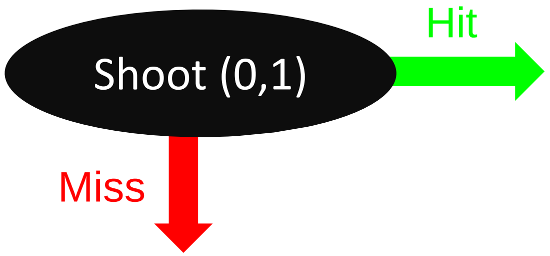


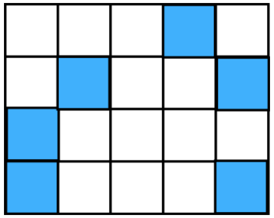
Shape

Shoot (0,1)

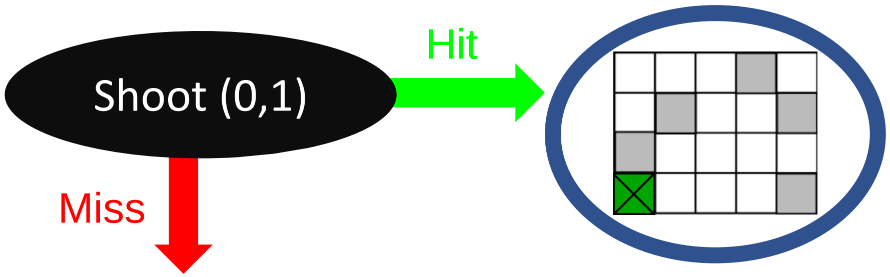


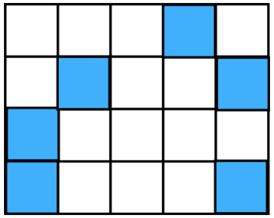
Shape





Shape

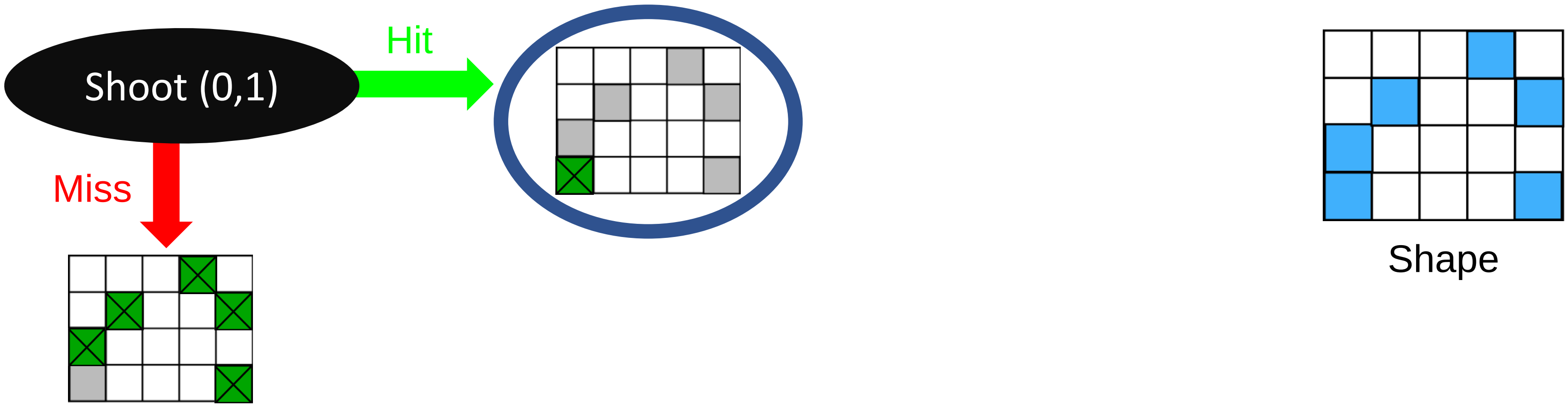


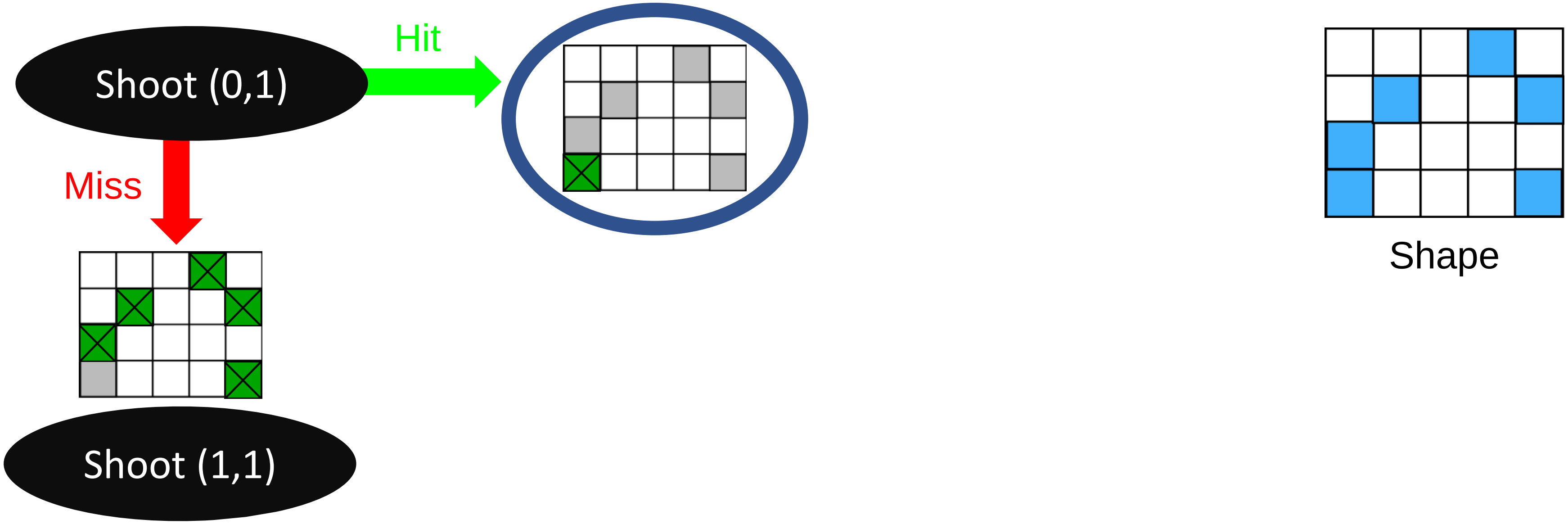


Shape

Shooting Algorithms

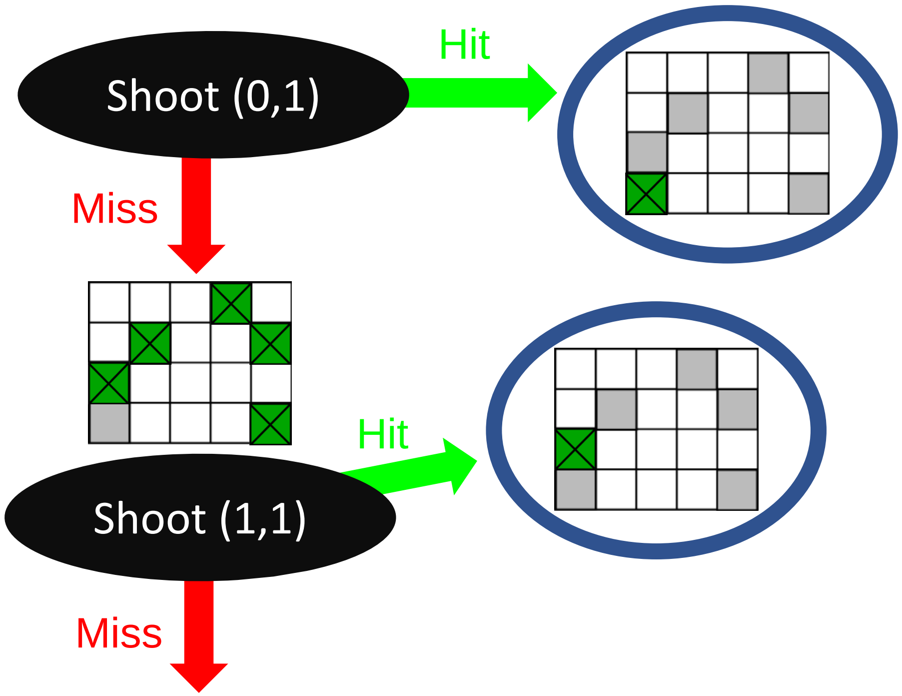
Example 4

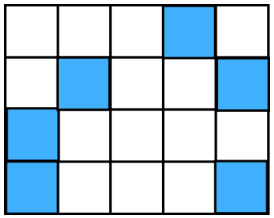



Shoot (0,1)
Hit
Miss
Shoot (1,1)
Shape

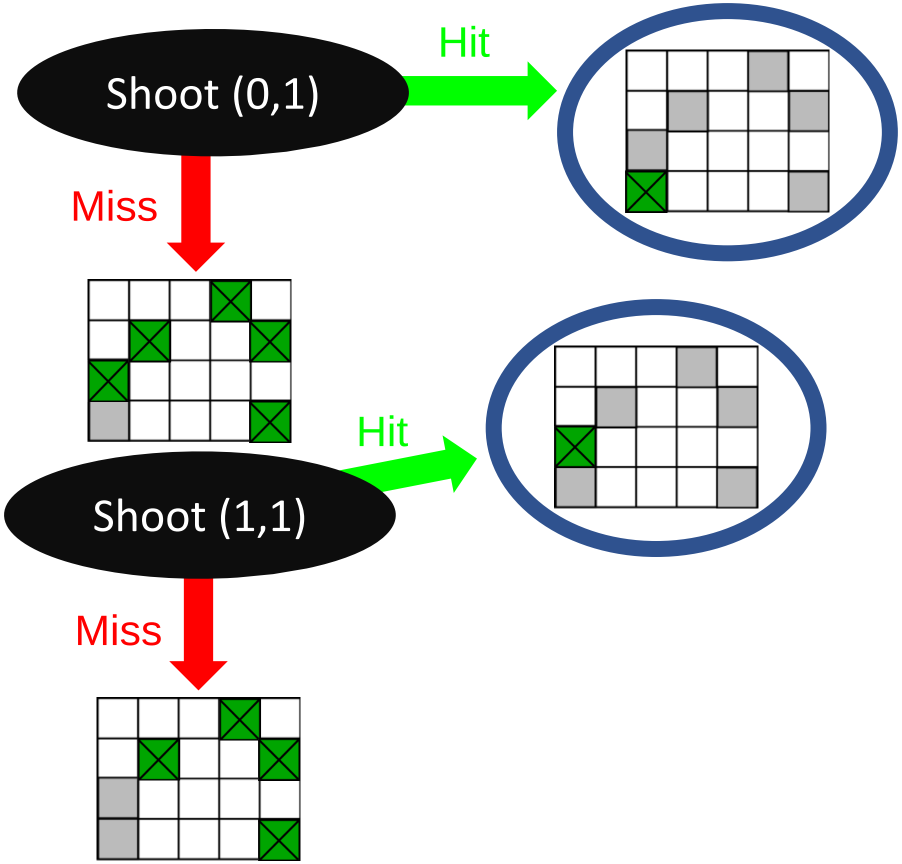
Shooting Algorithms

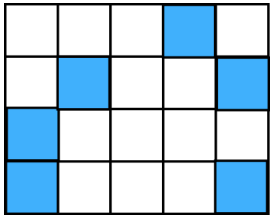
Example 4



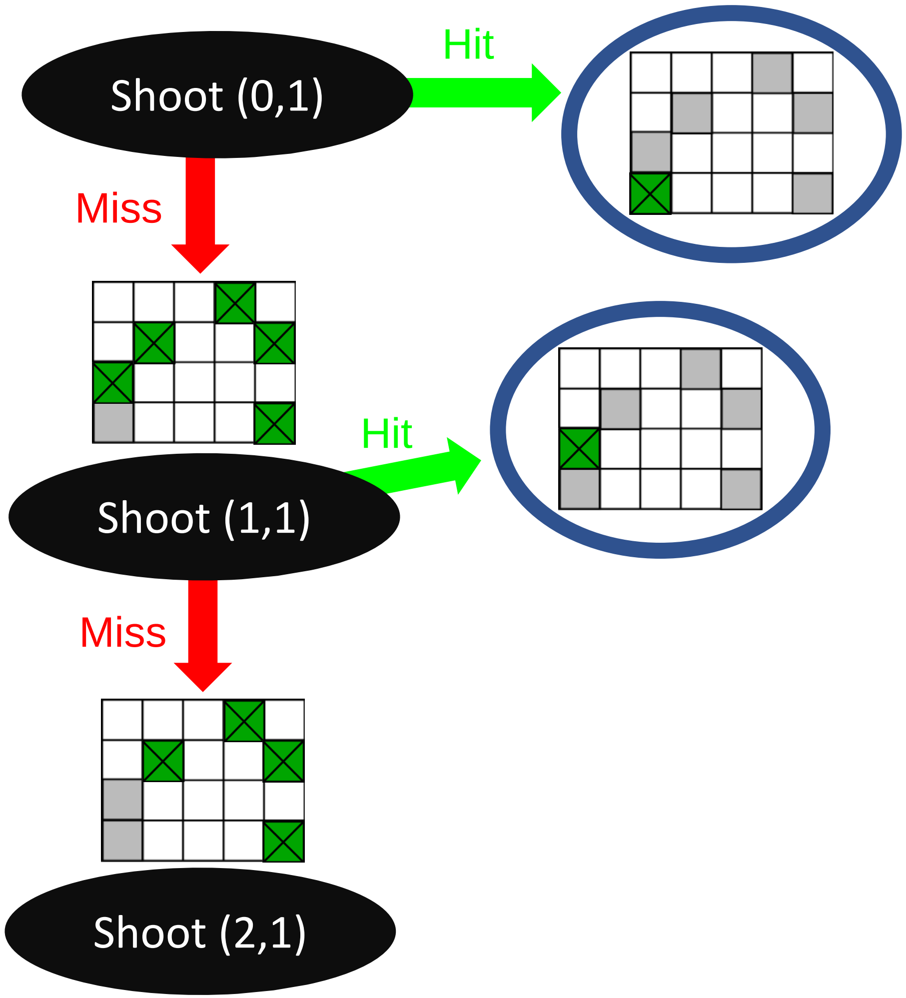


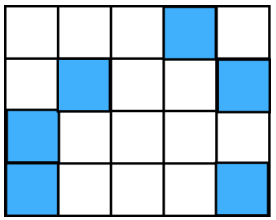
Shape



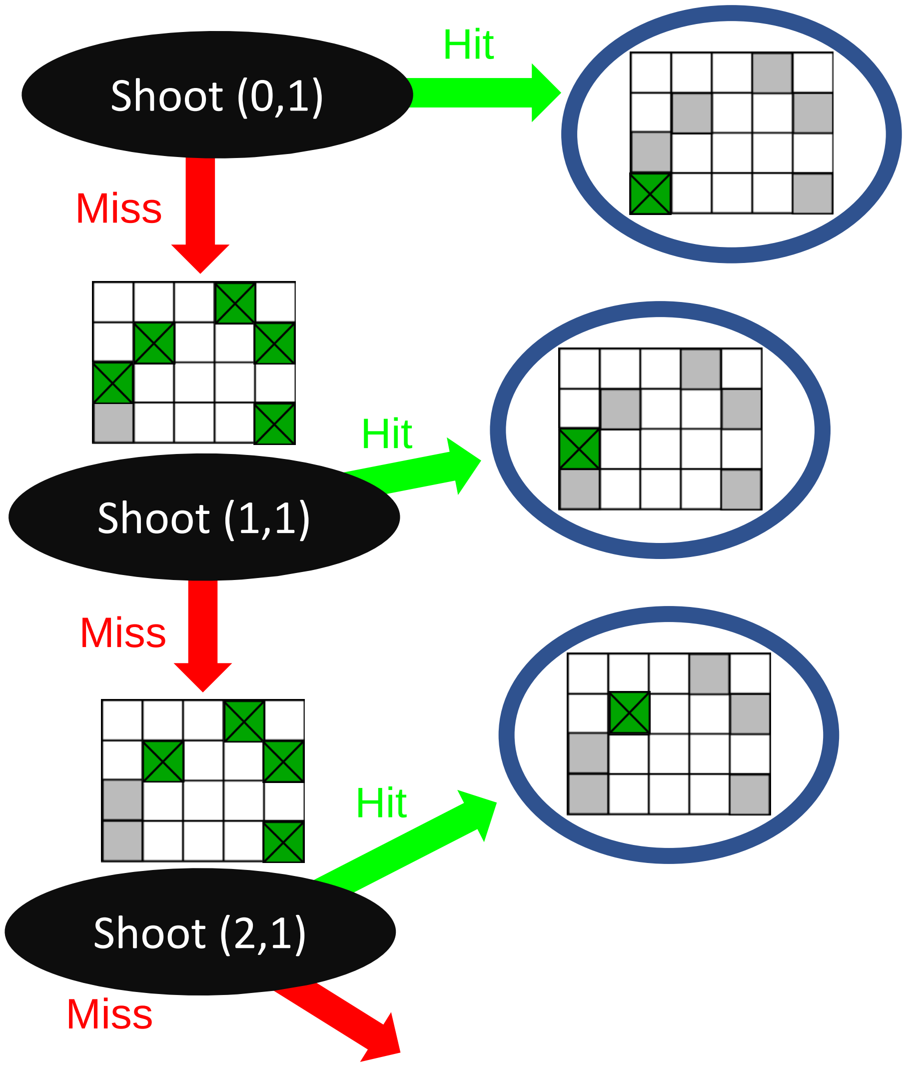


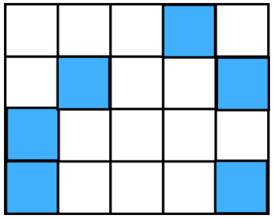
Shape




Shape

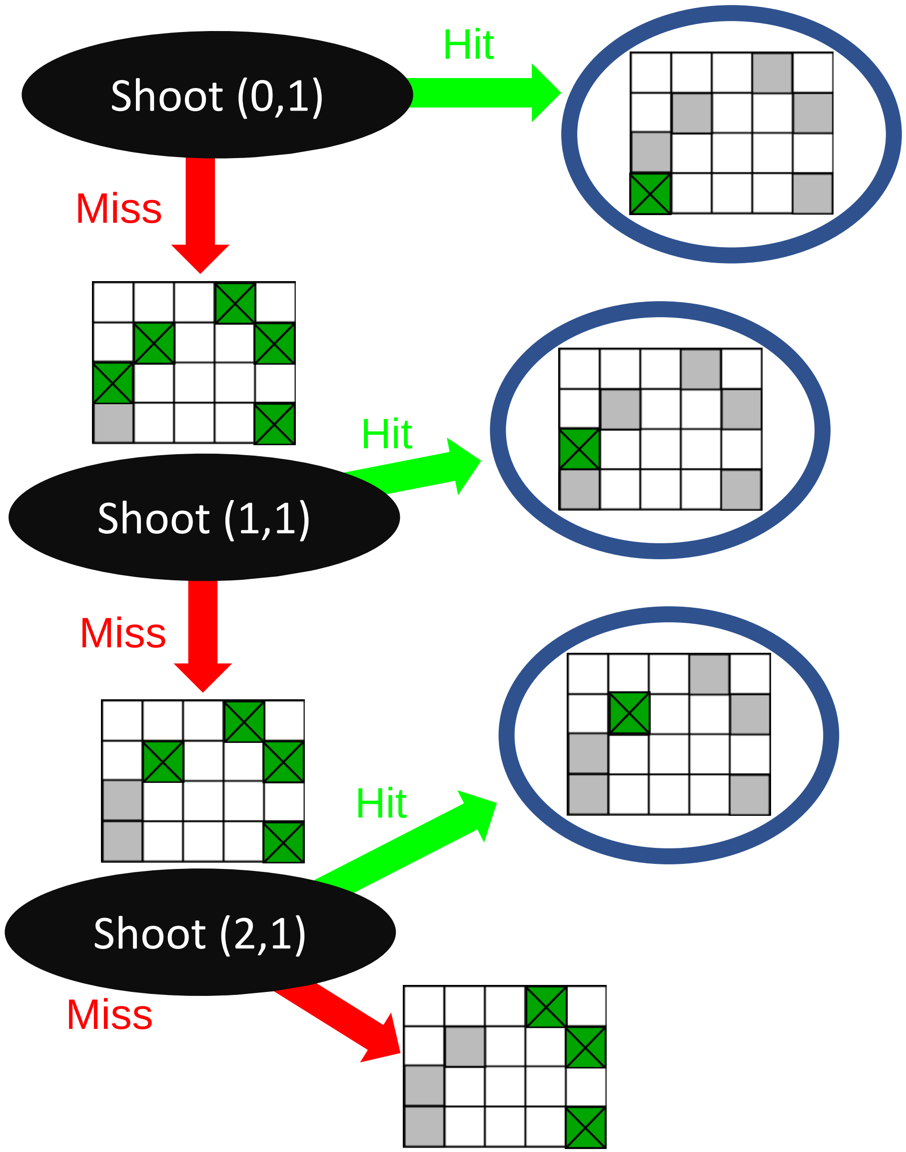
Example 4

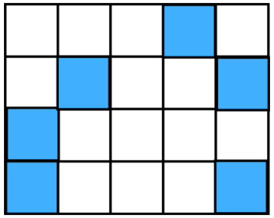




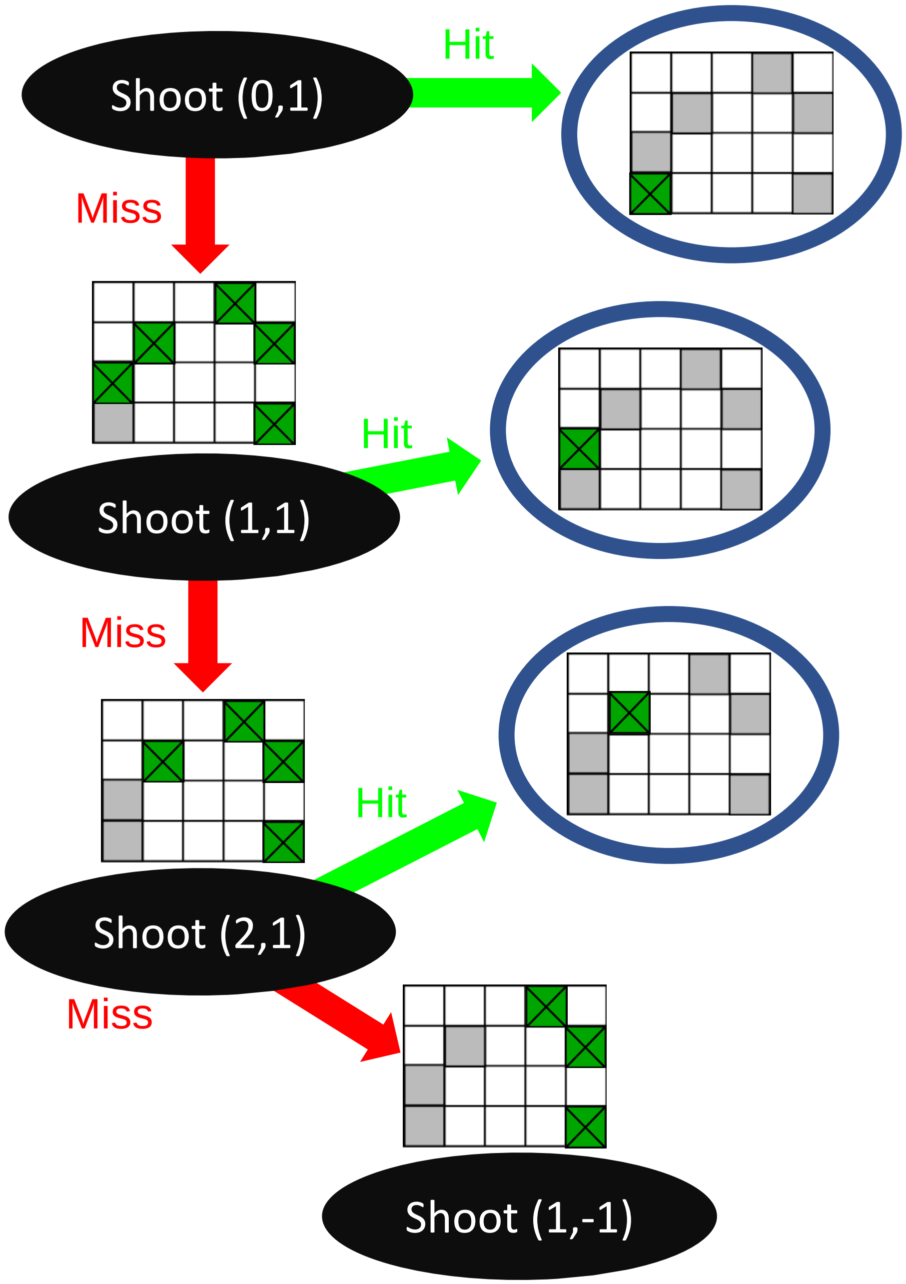
Shape

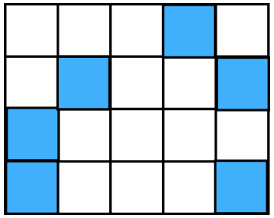
Another example

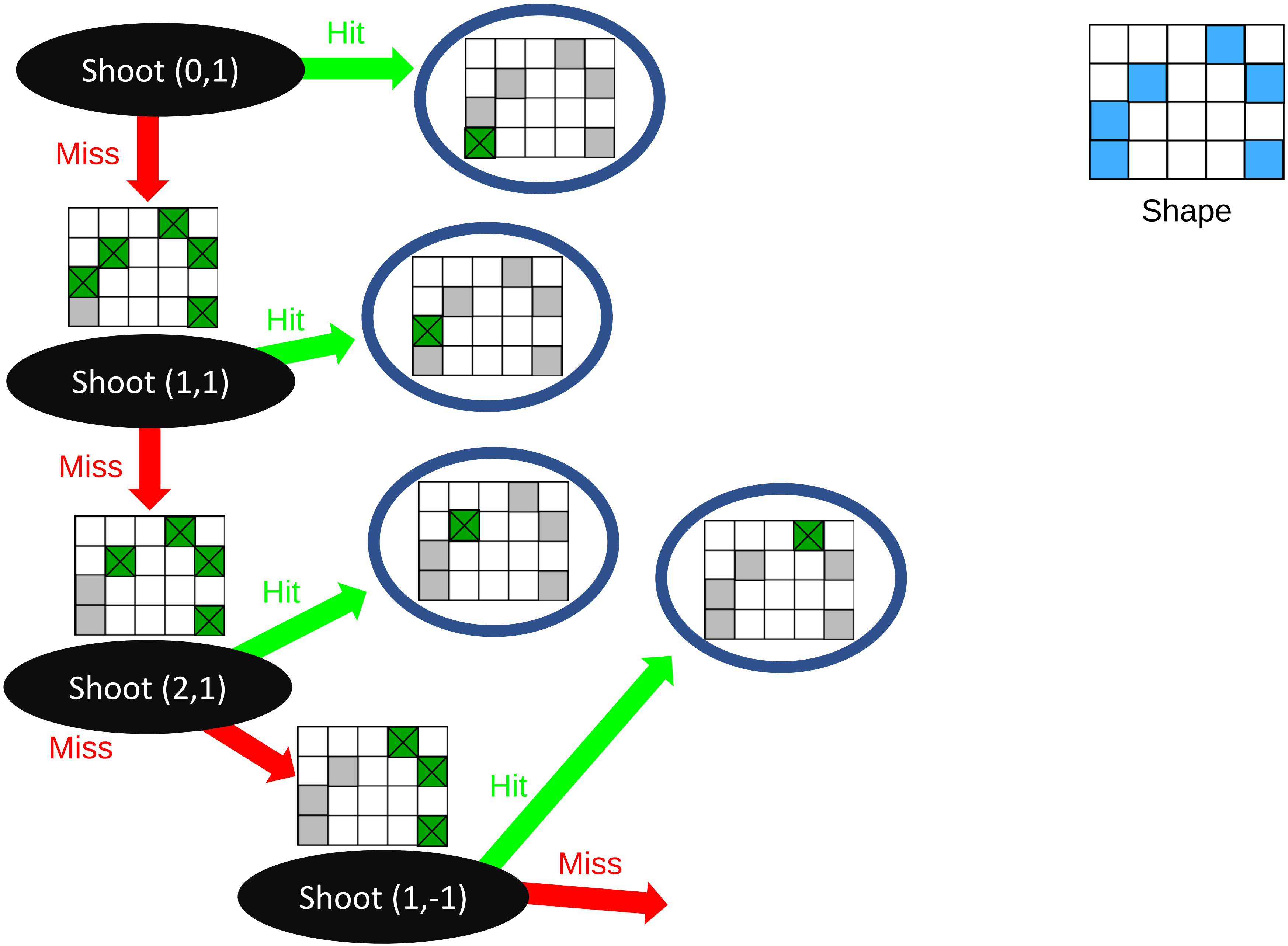




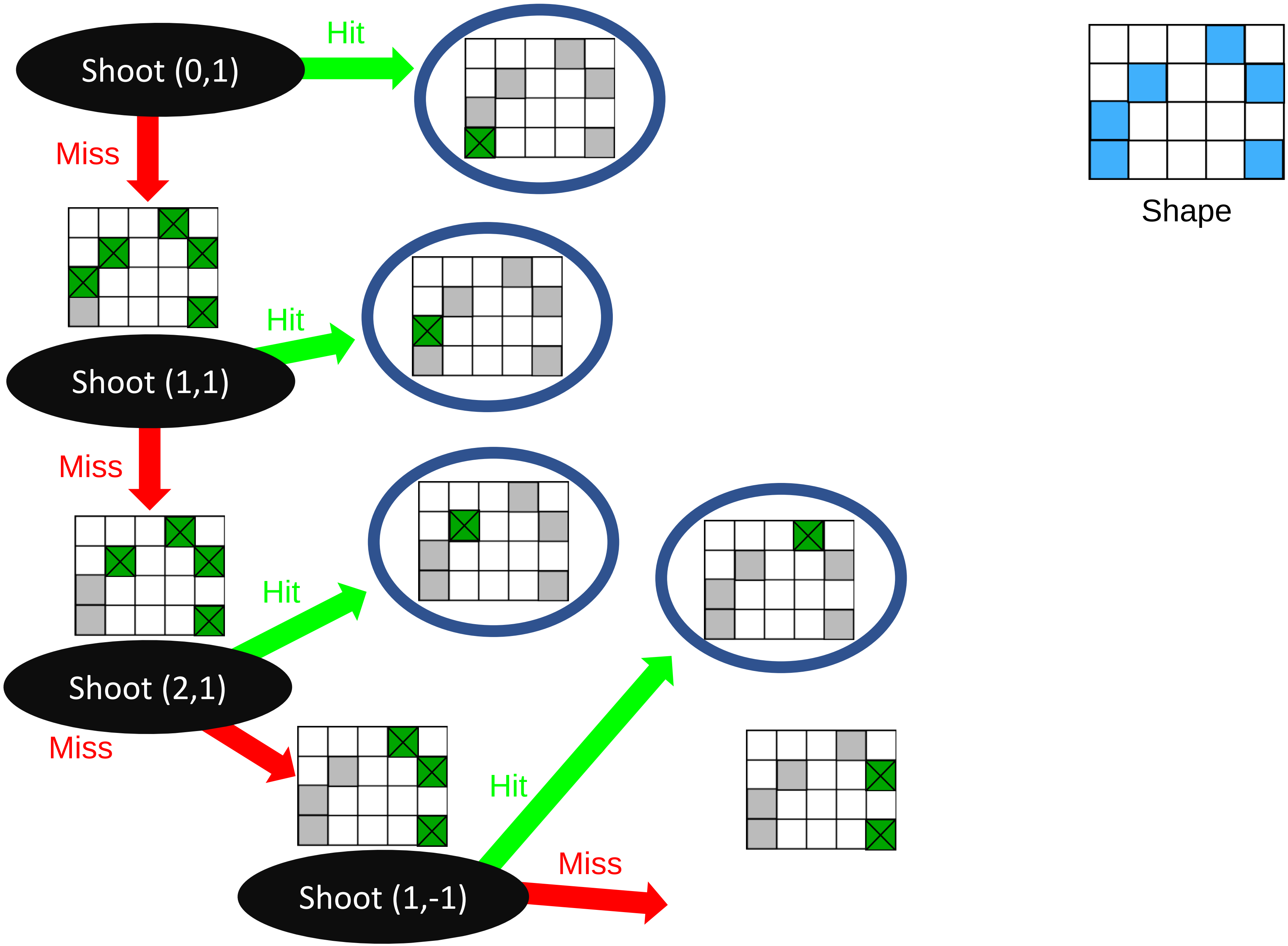
Shape

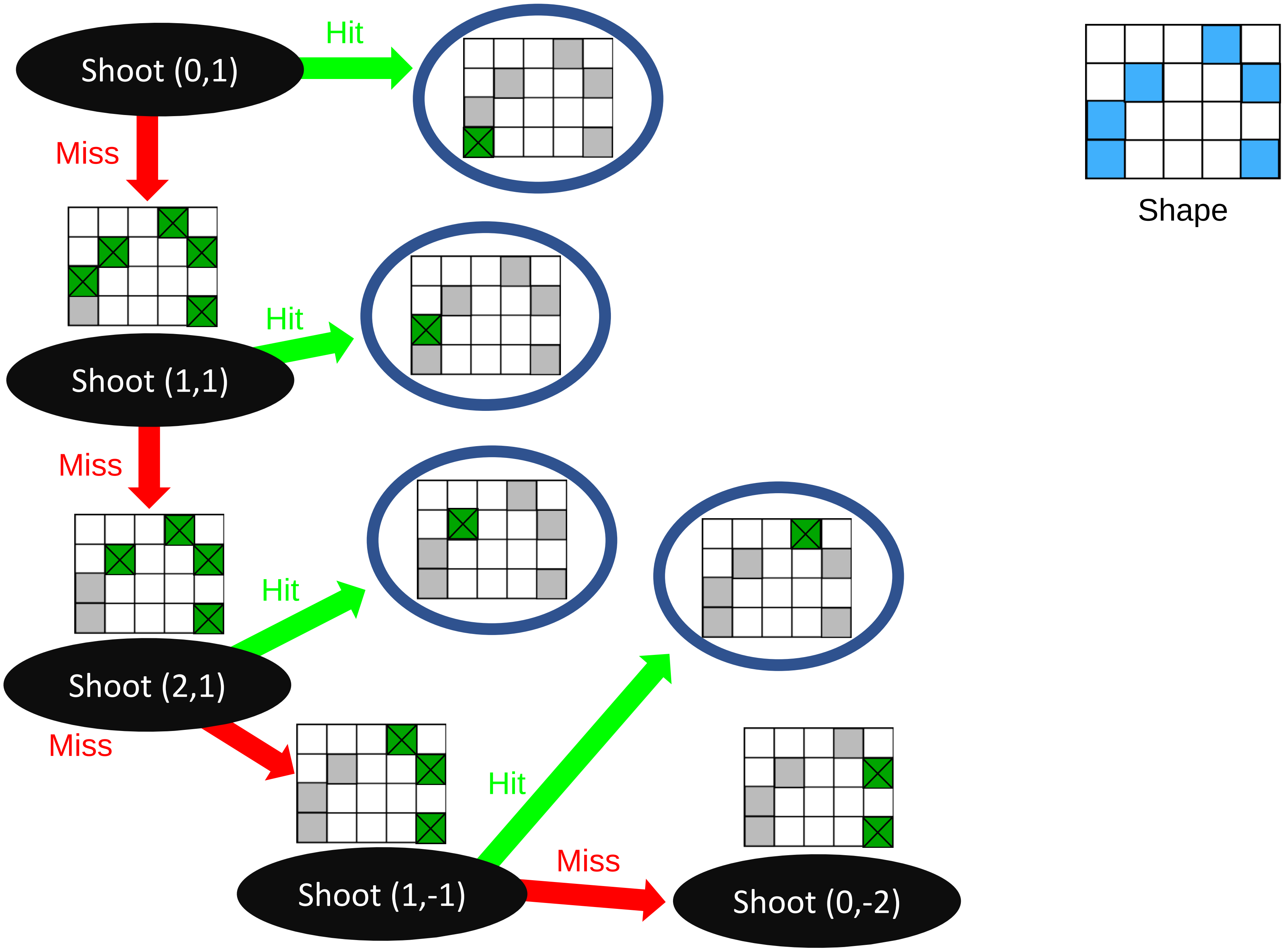



Shape

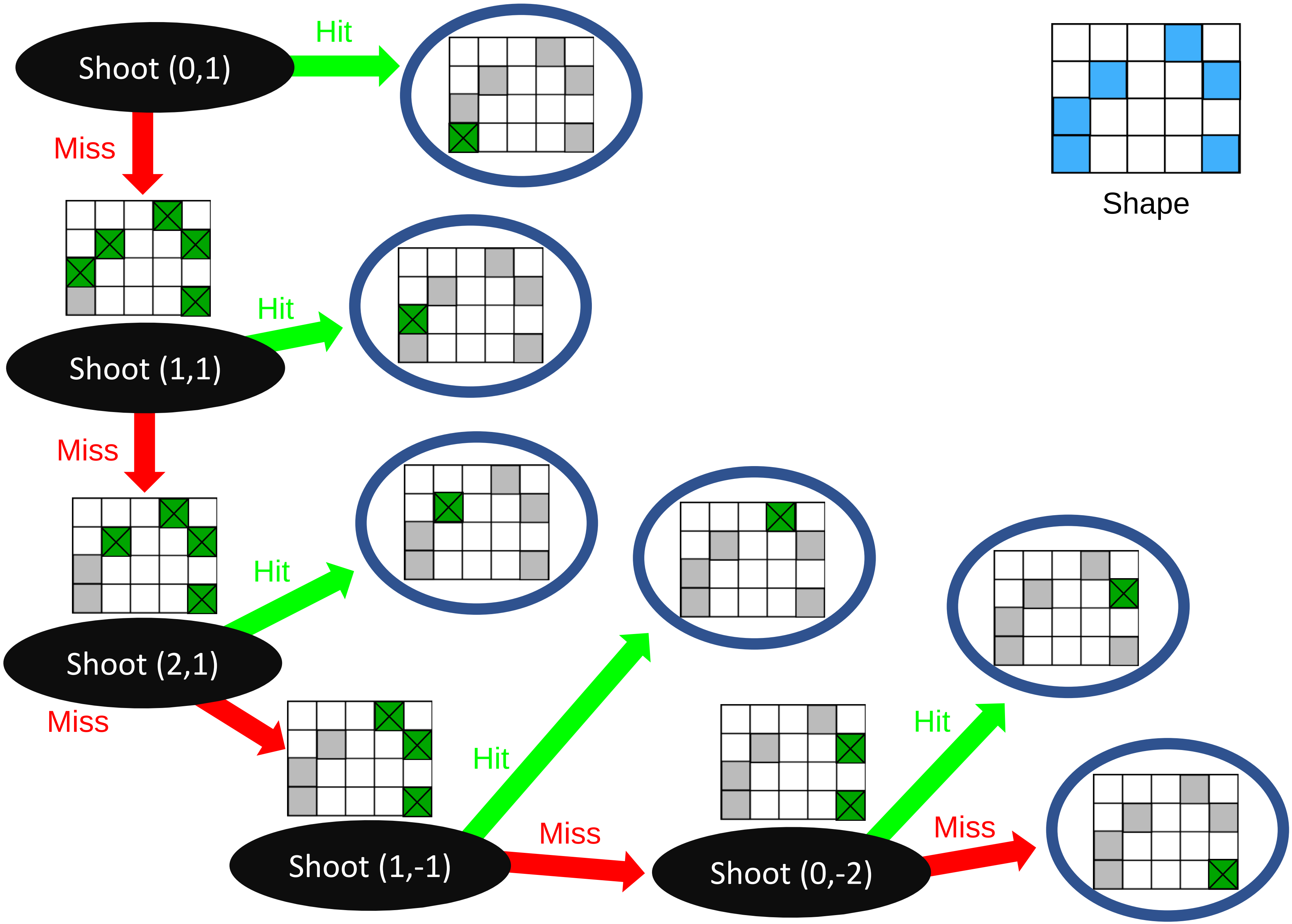

Shoot (0,1)
Hit
Miss
Shoot (1,1)
Hit
Miss
Shoot (2,1)
Hit
Miss
Shoot (1,-1)
Hit
Miss
Shape

Example 4


Shoot (0,1)
Hit
Miss
Shoot (1,1)
Hit
Miss
Shoot (2,1)
Hit
Miss
Shoot (1,-1)
Hit
Miss
Shape

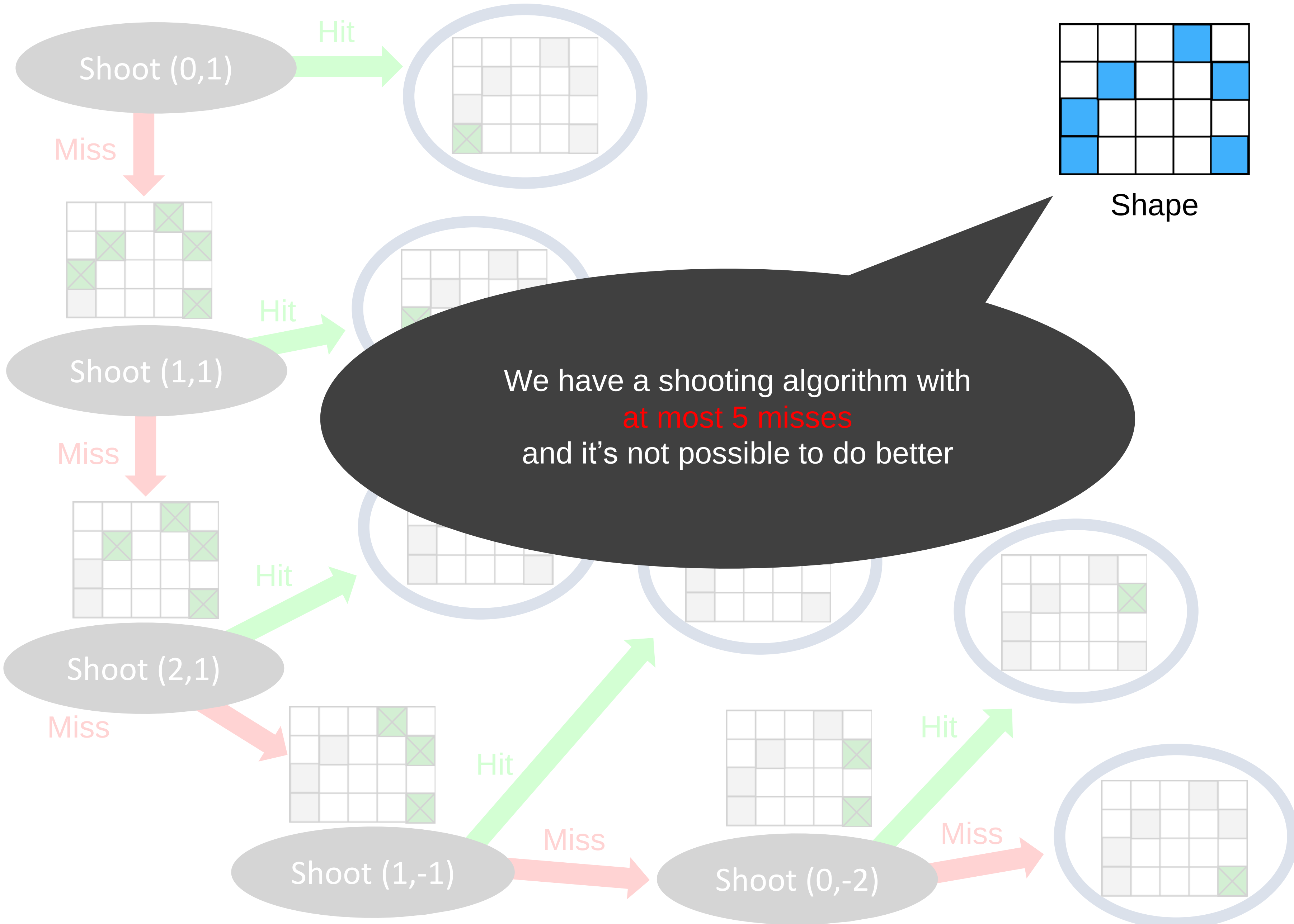

Shoot (0,1)
Hit
Miss
Shoot (1,1)
Hit
Miss
Shoot (2,1)
Hit
Miss
Shoot (1,-1)
Hit
Miss
Shoot (0,-2)
Shape

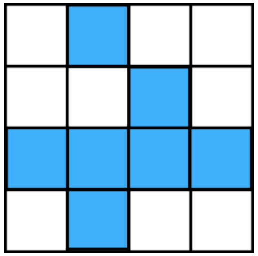
Example 4


Shoot (0,1)
Hit
Miss
Shoot (1,1)
Hit
Miss
Shoot (2,1)
Hit
Miss
Shoot (1,-1)
Hit
Miss
Shoot (0,-2)
Hit
Miss
Shape

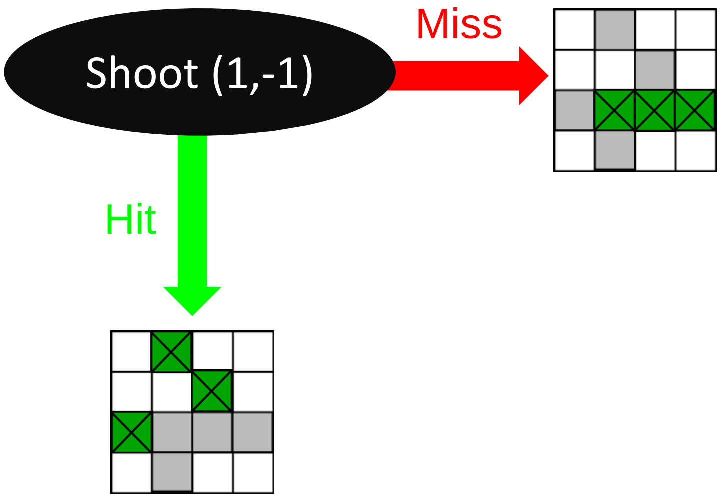
II Shooting Algorithms

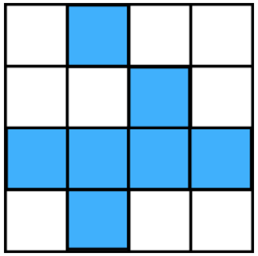
Example 4



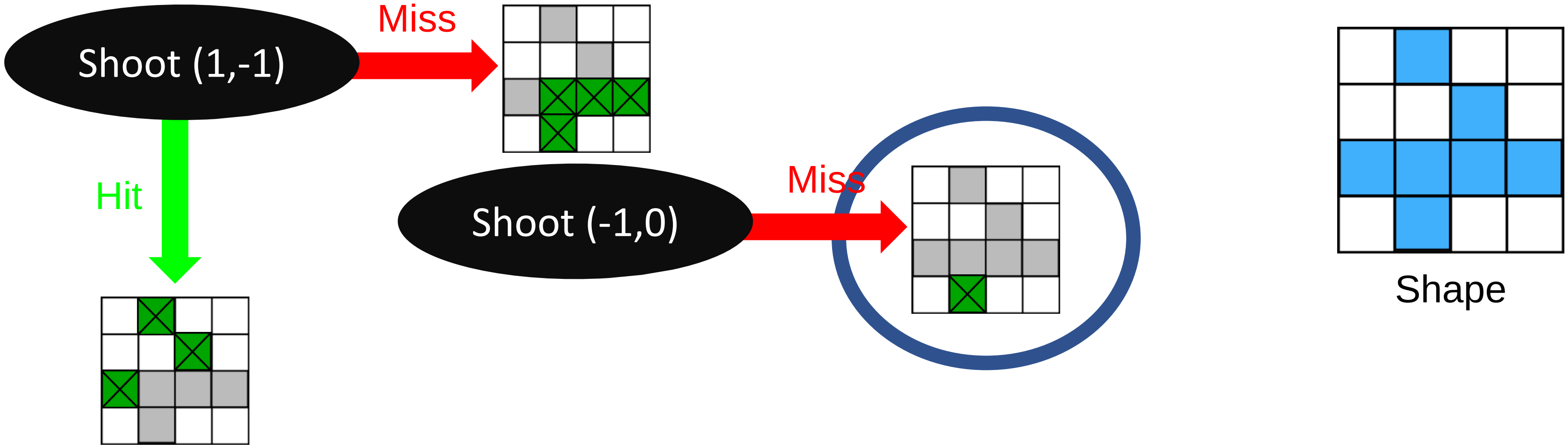


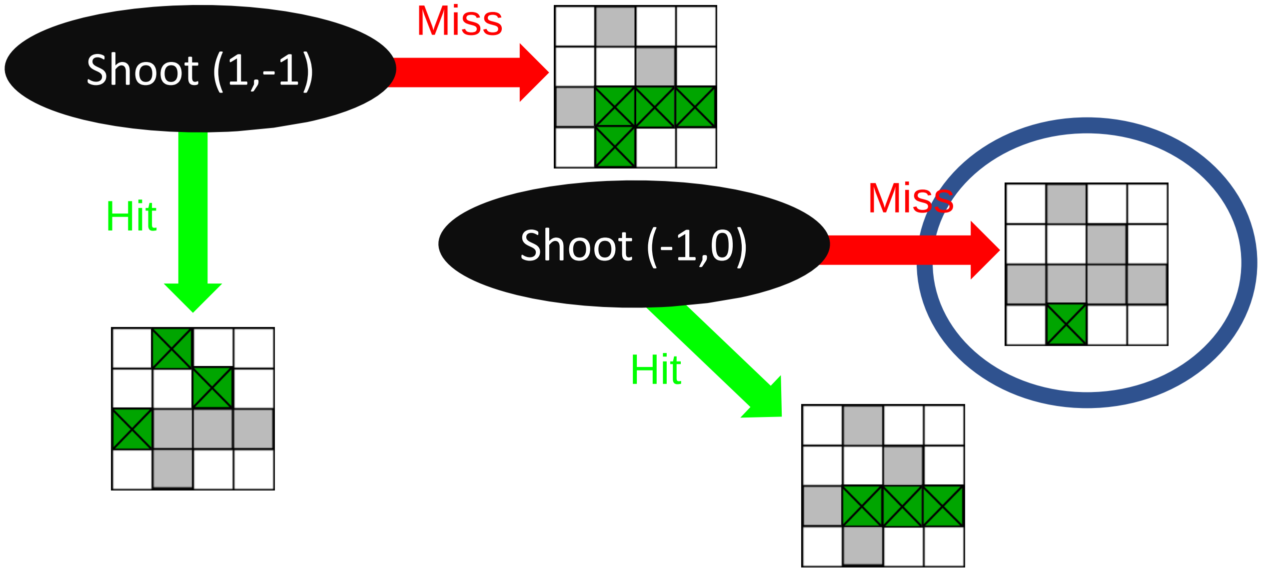
Shape

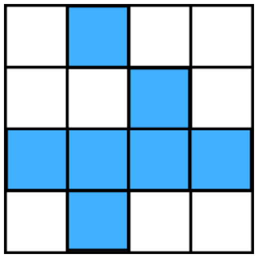


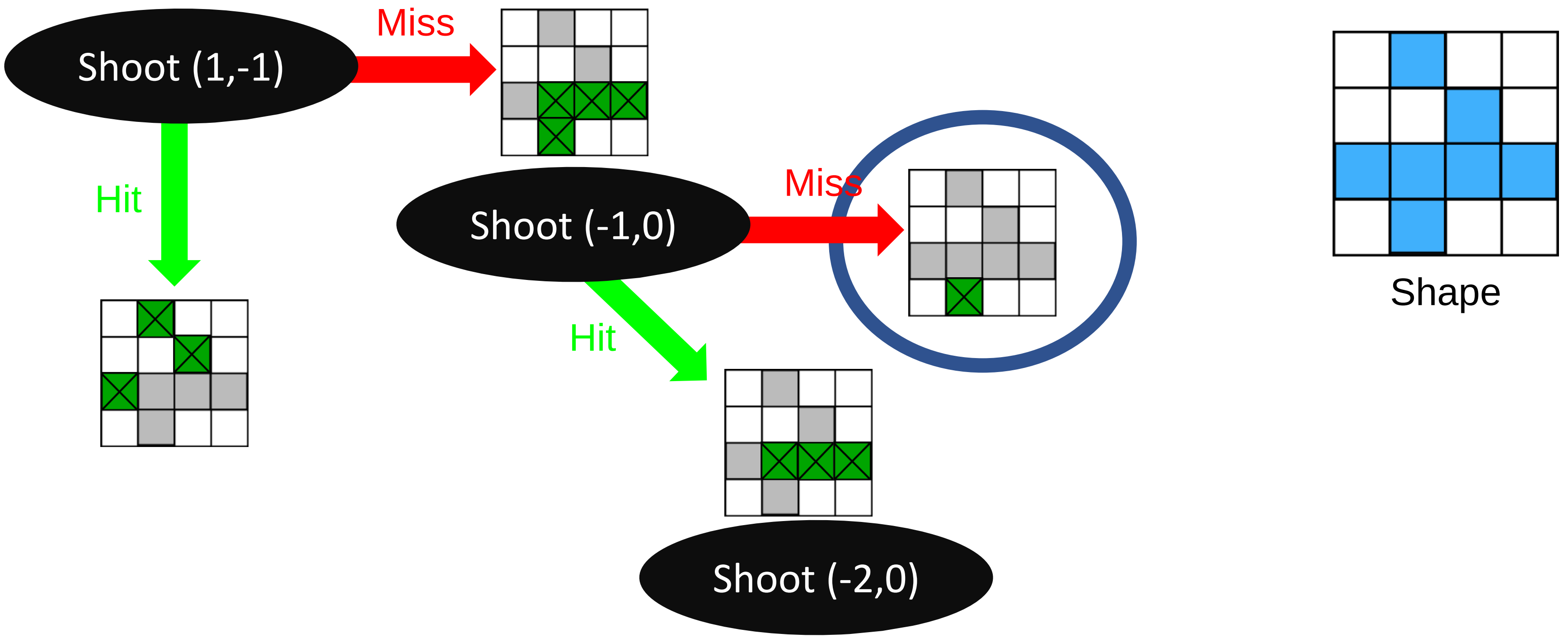


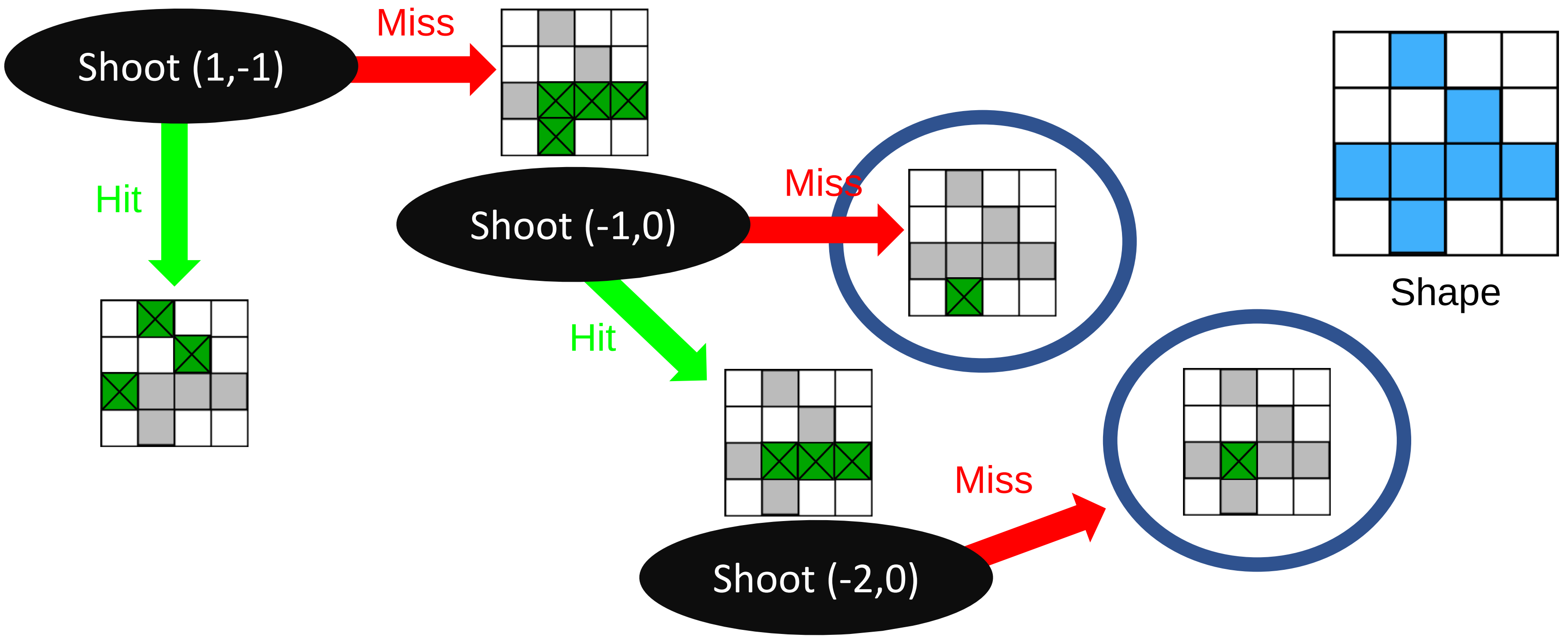
Shape

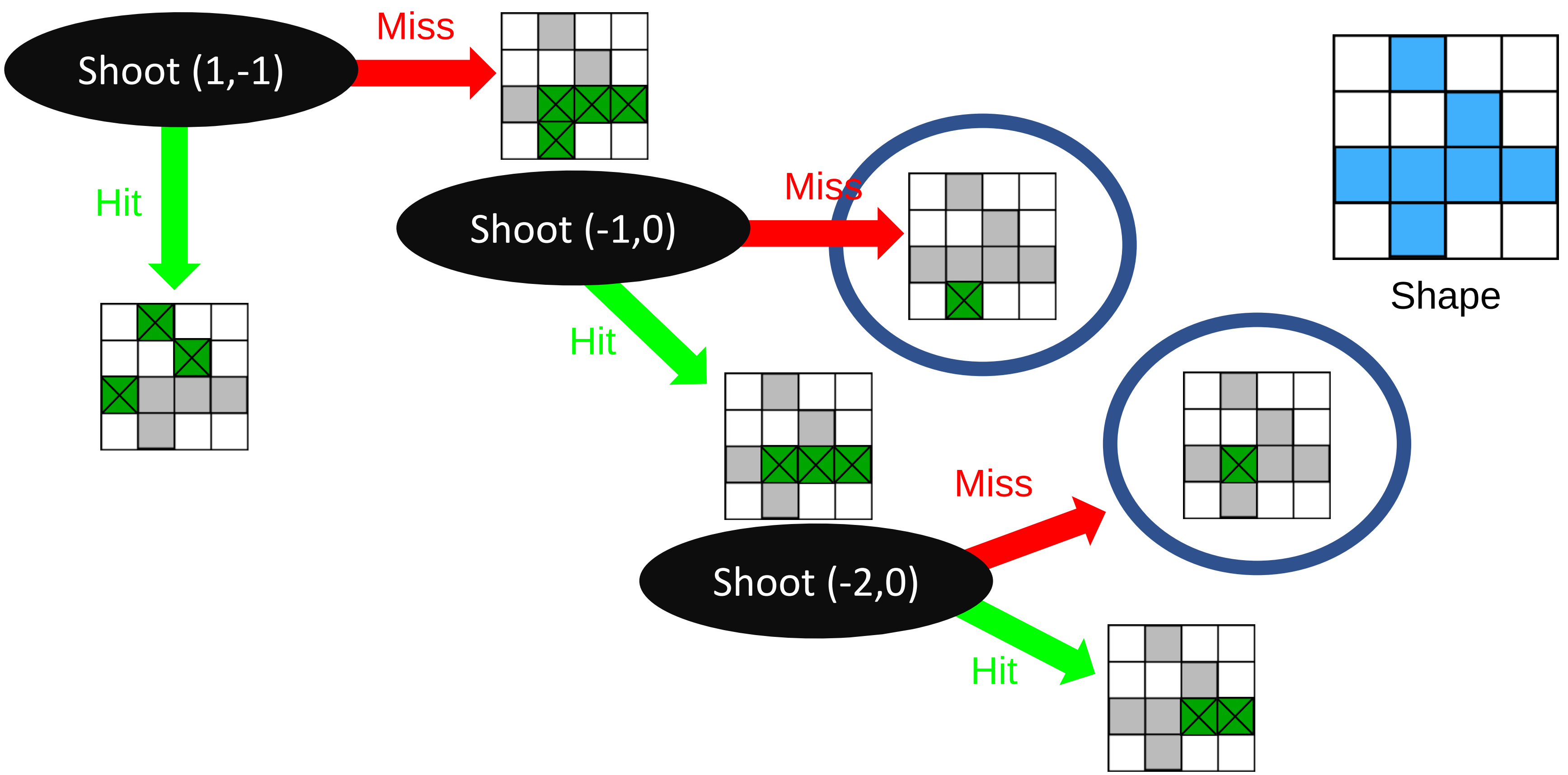

Miss
Shoot (1,-1)
Hit
Shoot (-1,0)
Miss
Shape

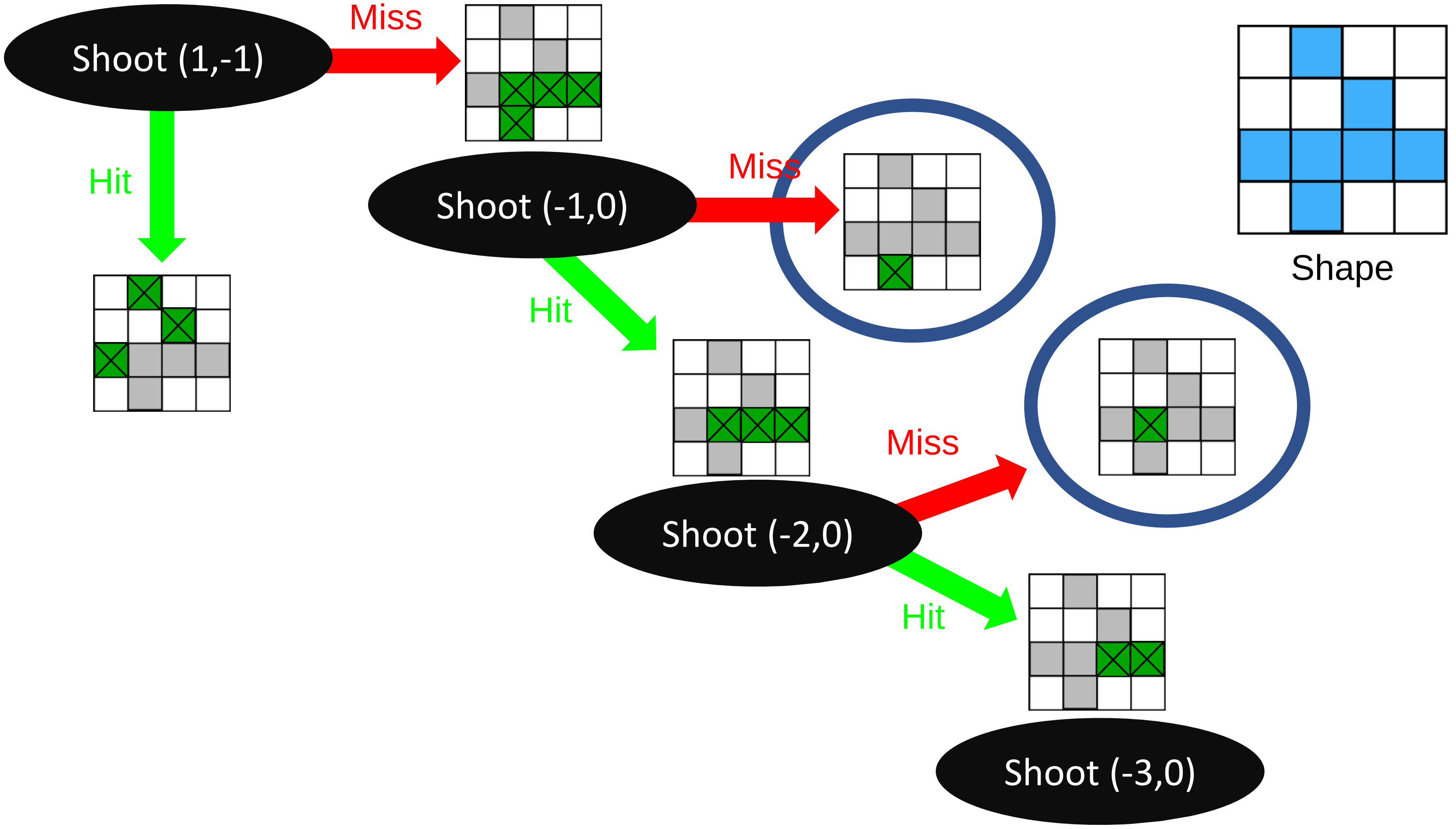



Shape

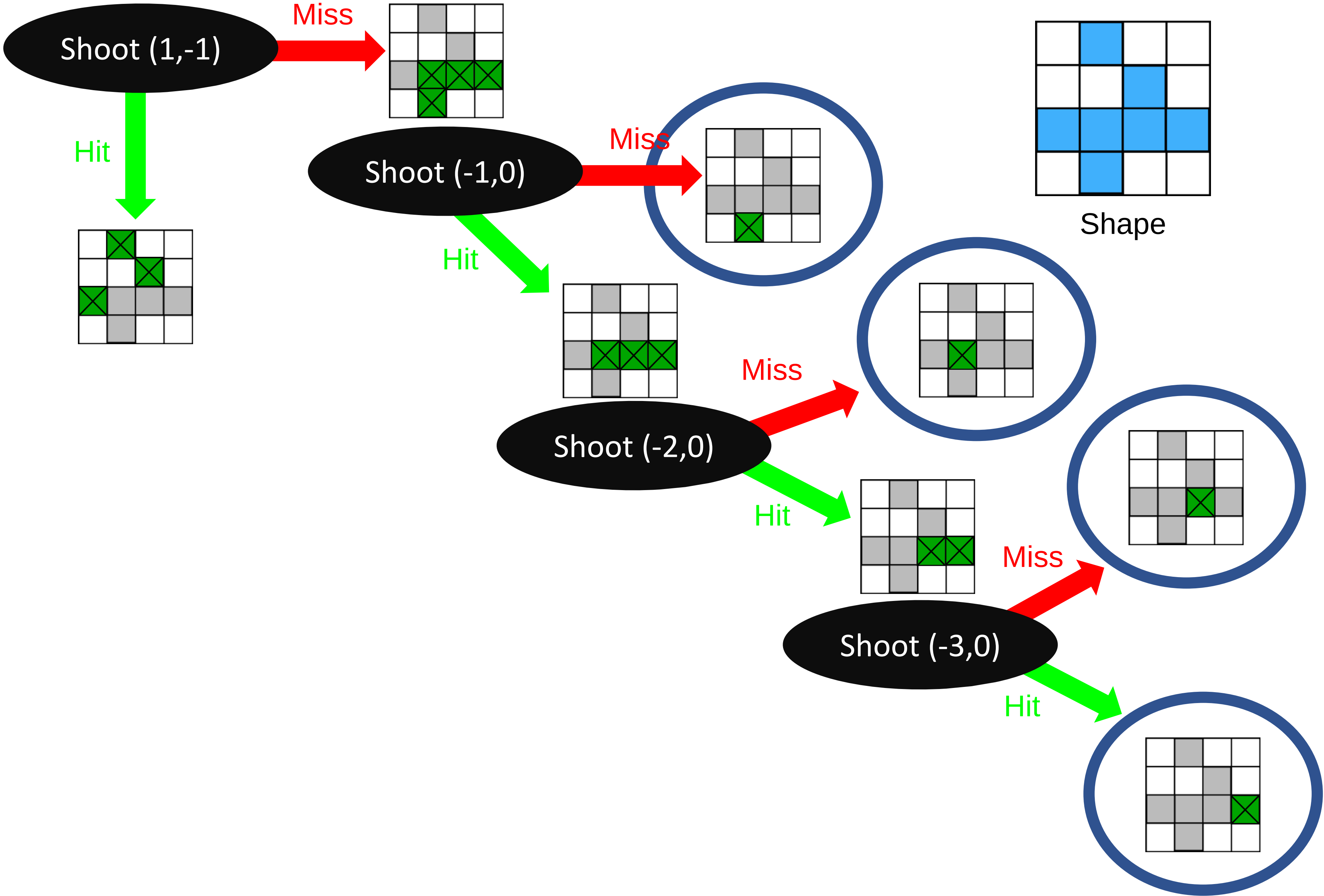

Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shoot (-2,0)
Shape

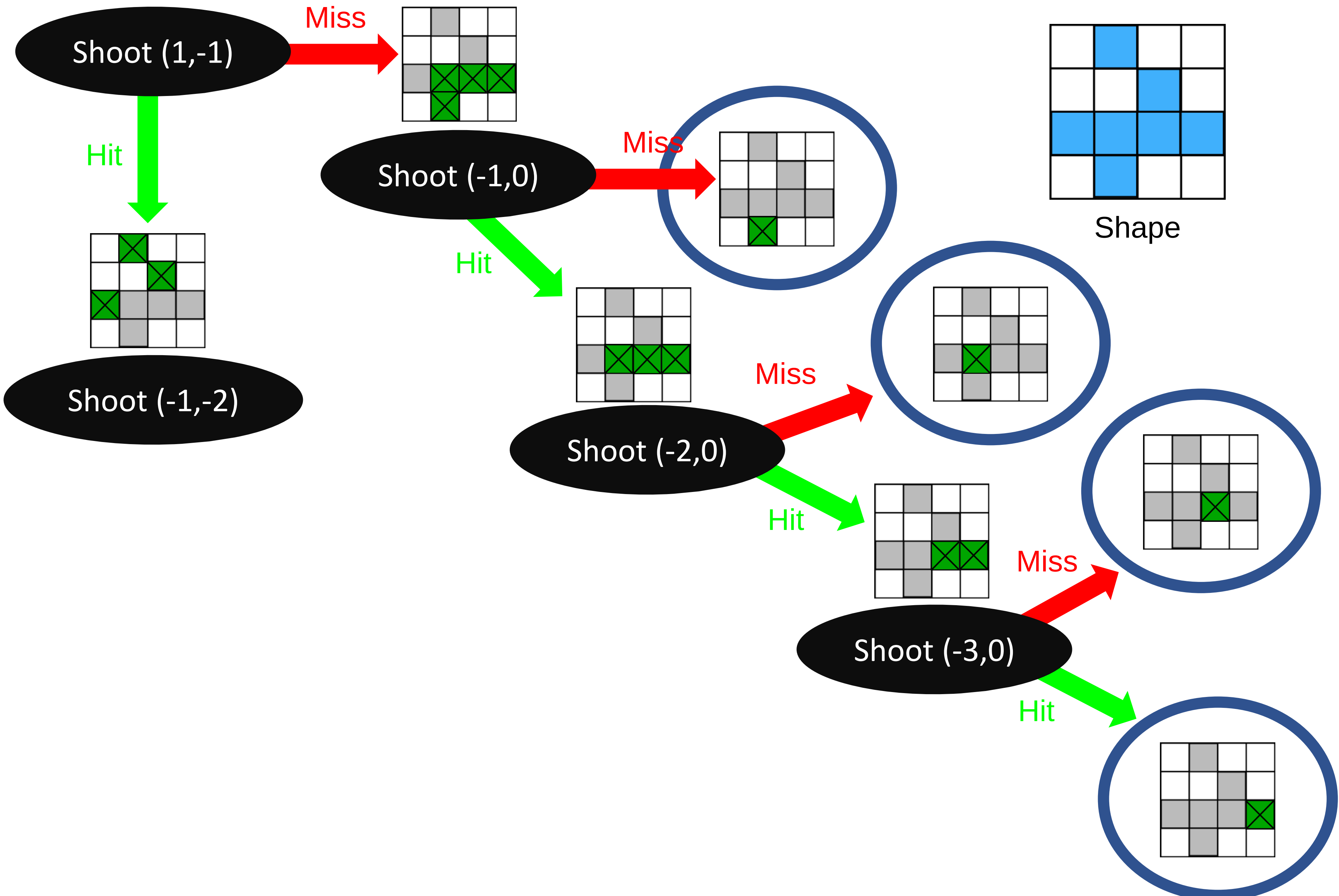

Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shoot (-2,0)
Miss
Shape

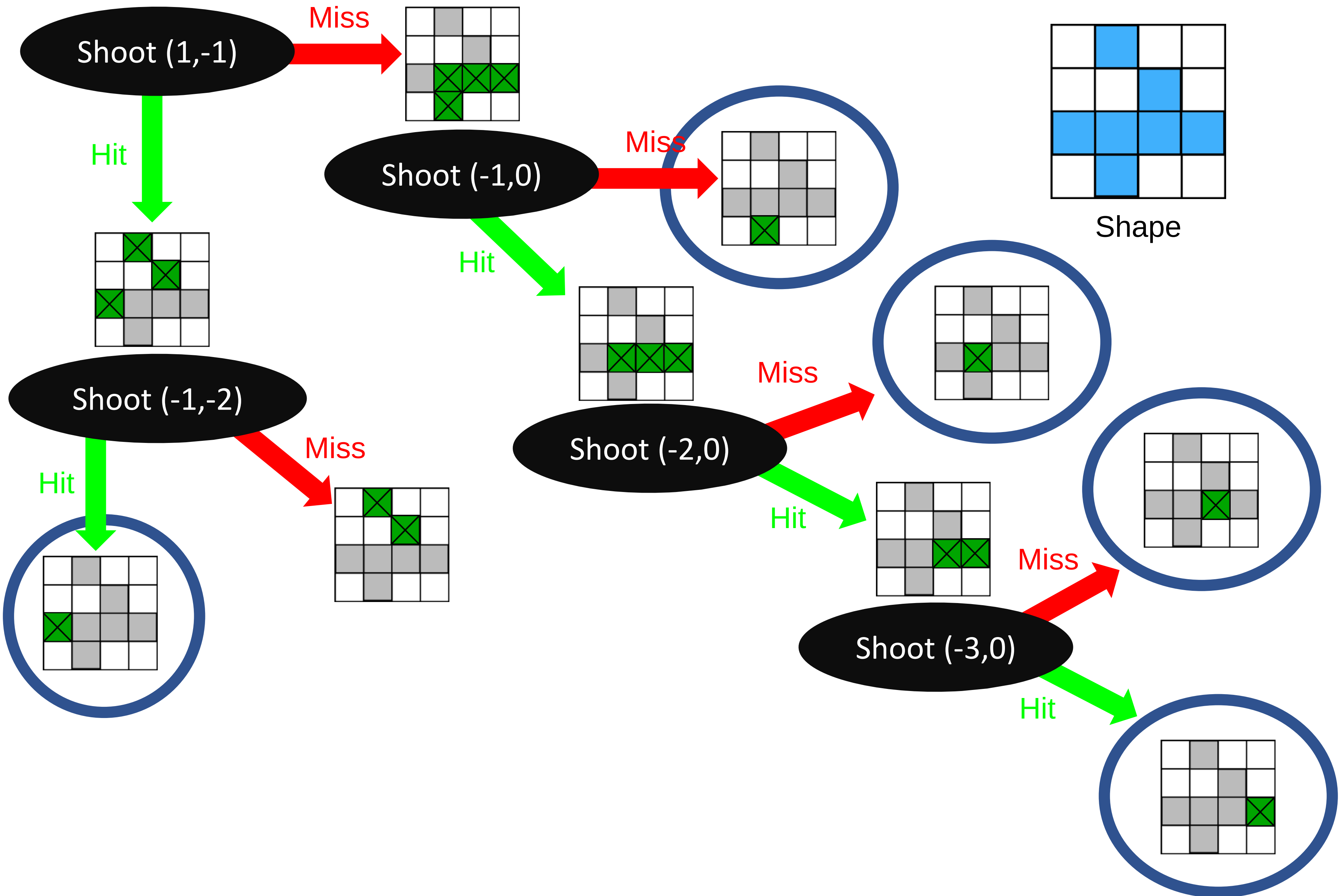

Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shoot (-2,0)
Miss
Hit
Shape

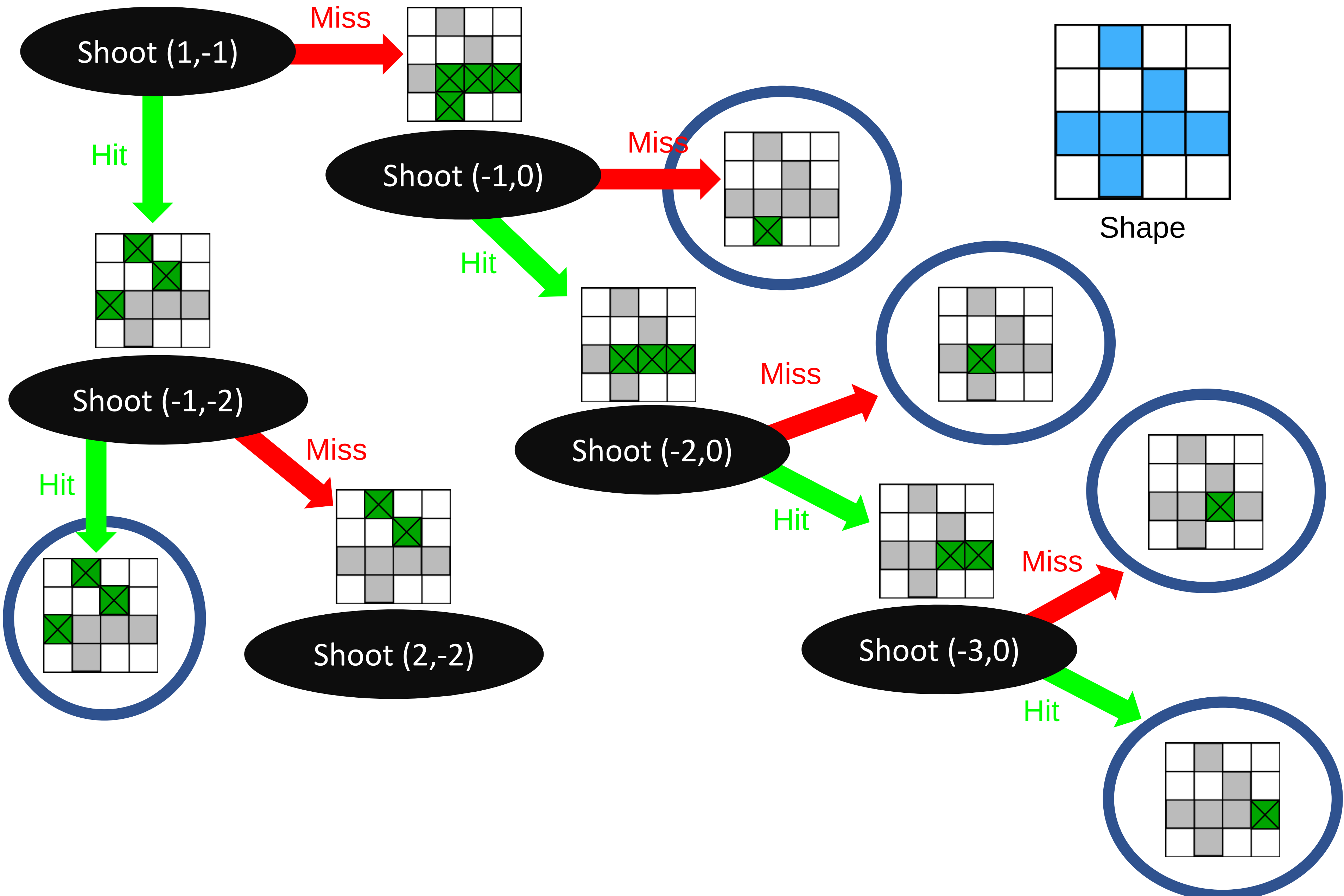

Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shoot (-2,0)
Miss
Hit
Shoot (-3,0)
Shape

Example 5

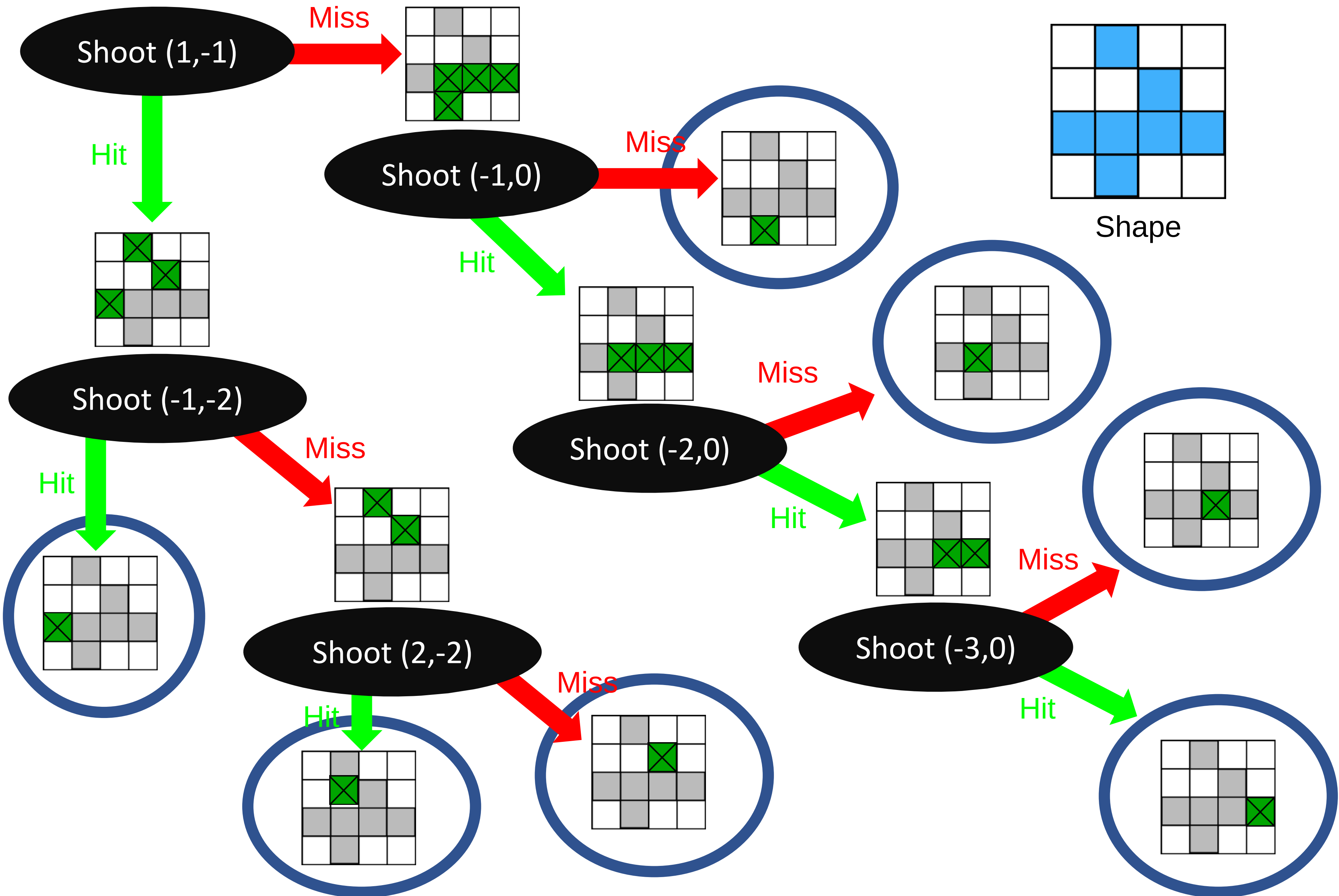

Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shoot (-2,0)
Miss
Hit
Shoot (-3,0)
Miss
Hit
Shape


Shoot (1,-1)
Miss
Hit
Shoot (-1,-2)
Shoot (-1,0)
Miss
Hit
Shape
Shoot (-2,0)
Miss
Hit
Shoot (-3,0)
Miss
Hit

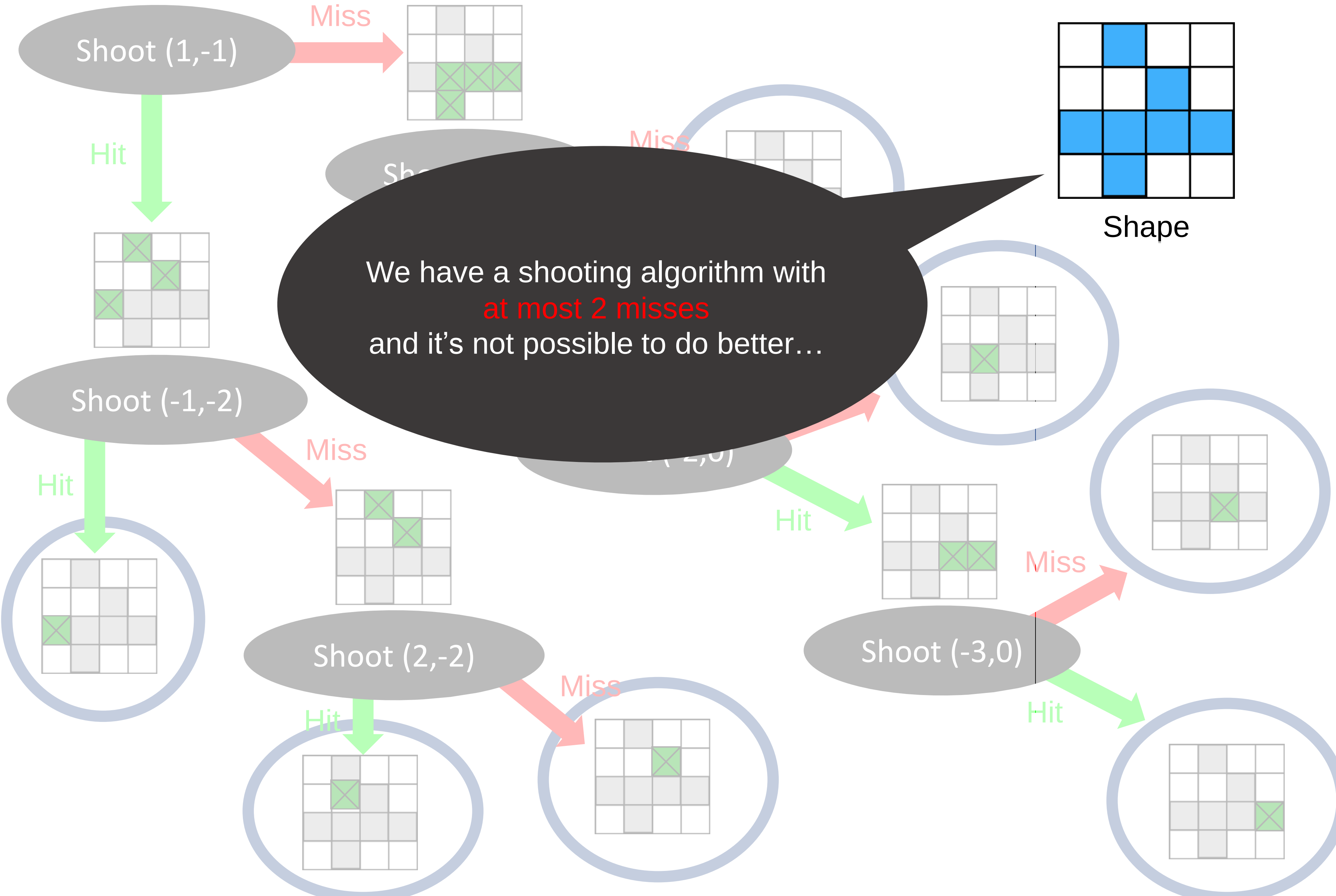

Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shape
Shoot (-1,-2)
Hit
Miss
Shoot (-2,0)
Miss
Hit
Shoot (-3,0)
Miss
Hit


Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shoot (-1,-2)
Hit
Miss
Shoot (2,-2)
Shoot (-2,0)
Miss
Hit
Shoot (-3,0)
Miss
Hit
Shape

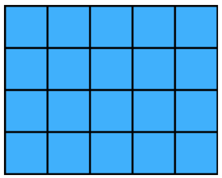
Example 5



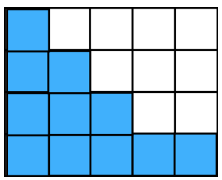
Shoot (1,-1)
Miss
Hit
Shoot (-1,0)
Miss
Hit
Shape
Shoot (-1,-2)
Hit
Miss
Shoot (-2,0)
Miss
Hit
Shoot (2,-2)
Hit
Miss
Shoot (-3,0)
Miss
Hit


Shoot (1,-1)
Miss
Hit
Shoot (-1,-2)
Miss
Hit
Shoot (2,-2)
Hit
Miss
Hit
Shoot (-3,0)
Miss
Hit
Shape
We have a shooting algorithm with
at most 2 misses
and it's not possible to do better…

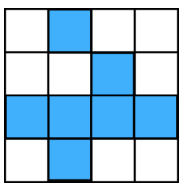
We have an algorithm with at most 1 miss !



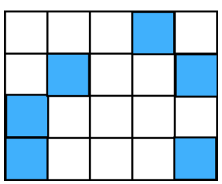
We have an algorithm with at most 2 misses !



We have an algorithm with at most 2 misses !

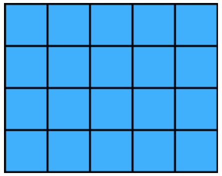


We have an algorithm with at most 2 misses !

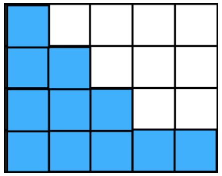


We have an algorithm with at most 5 misses !

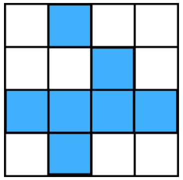
We have an algorithm with at most 1 miss !



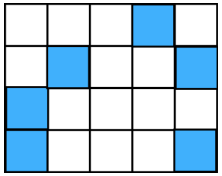
We have an algorithm with at most 2 misses !



We have an algorithm with at most 2 misses !



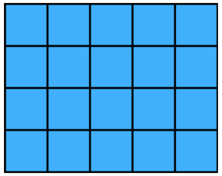
We have an algorithm with at most 2 misses !



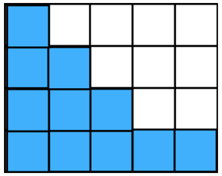
We have an algorithm with at most 5 misses !

And we cannot do better…

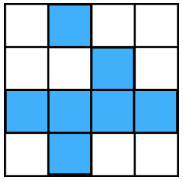
We have an algorithm with at most 1 miss !



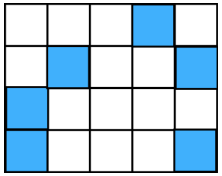
We have an algorithm with at most 2 misses !



We have an algorithm with at most 2 misses !



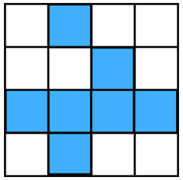
We have an algorithm with at most 2 misses !



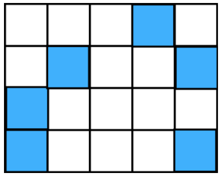
We have an algorithm with at most 5 misses !

And we cannot do better…

This integer is the *battleship complexity* of each shape



We have an algorithm with at most 2 misses !



We have an algorithm with at most 5 misses !

And we cannot do better…

This integer is the *battleship complexity* of each shape

Definition

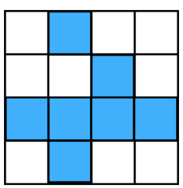
The *battleship complexity* of a shape is a minimum of a maximum:

it is the minimum among all shooting algorithms

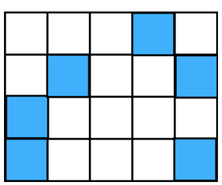
of the maximum number of misses that we can get with this algorithm

among all positions of the ship.

Complexity (S)= $\mathrm{Min}_{\text{shooting algorithm}} (\mathrm{Max}_{\text{ship positions}} (\text{number of misses}))$



We have an algorithm with at most 2 misses !



We have an algorithm with at most 5 misses !

And we cannot do better…

This integer is the *battleship complexity* of each shape

Plan

I Problem Statement

II Shooting Algorithms

III Results

IV Proofs

Plan

I Problem Statement

II Shooting Algorithms

III Results

IV Proofs

Definition

The *battleship complexity* of a shape is a minimum of a maximum:

it is the minimum among all shooting algorithms

of the maximum number of misses that we can get with this algorithm

among all positions of the ship.

Complexity(S)= $\mathrm{Min}_{\text{shooting algorithm}}\ (\mathrm{Max}_{\text{ship positions}}\ (\text{number of misses}))$

Definition

The *battleship complexity* of a shape is a minimum of a maximum:

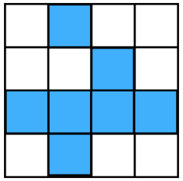
it is the minimum among all shooting algorithms

of the maximum number of misses that we can get with this algorithm

among all positions of the ship.

$$\text{Complexity}(S) = \text{Min}_{\text{shooting algorithm}} \left(\text{Max}_{\text{ship positions}} \left(\text{number of misses}\right)\right)$$

Is it easy to compute ?



Complexity= 2

Definition

The *battleship complexity* of a shape is a minimum of a maximum:

it is the minimum among all shooting algorithms

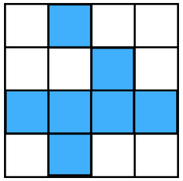
of the maximum number of misses that we can get with this algorithm

among all positions of the ship.

$$\text{Complexity}(S) = \text{Min}_{\text{shooting algorithm}} \left(\text{Max}_{\text{ship positions}} (\text{number of misses})\right)$$

Is it easy to compute ?

NO!!!!! (max for all shooting algorithms!)



Complexity= 2

Definition

The *battleship complexity* of a shape is a minimum of a maximum:

it is the minimum among all shooting algorithms

of the maximum number of misses that we can get with this algorithm

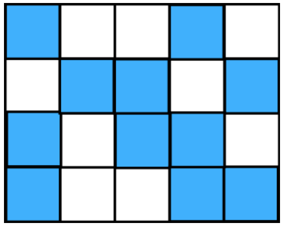
among all positions of the ship.

Complexity(S)= $\mathrm{Min}_{\text{shooting algorithm}}\,(\mathrm{Max}_{\text{ship positions}}\,(\text{number of misses}))$

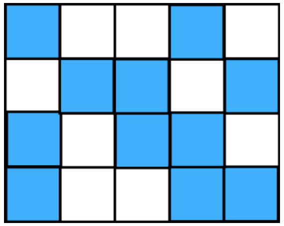
Is it easy to compute ?

NO!!!!! (max for all shooting algorithms!)

Any shooting algorithm provides an upper bound of the Battleship Complexity…



A shape *S* with *n* squares

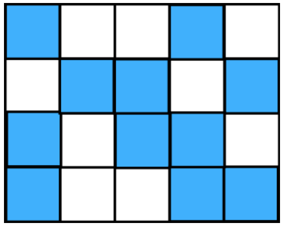


A shape *S* with *n* squares

Bound 1

For any shape

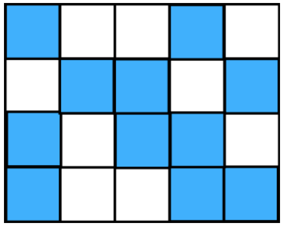
Complexity(S) ≤ n-1



A shape *S* with *n* squares

Bound 1
For any shape
Complexity(S) ≤ n-1

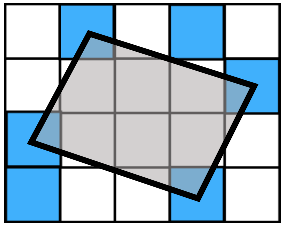
For parallelogram-free shapes, we have
Complexity(S)=n-1



A shape *S* with *n* squares

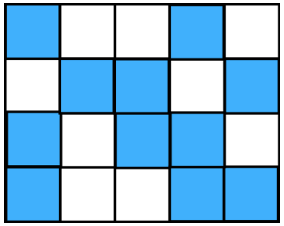
Bound 1
For any shape
Complexity(S) ≤ n-1

For parallelogram-free shapes, we have Complexity(S)=n-1



Non Parallelogram-free shape

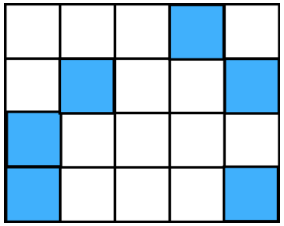
There exists 4 distinct points u,v,s,t in S with v-u=t-s



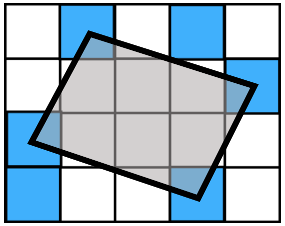
A shape *S* with *n* squares

Bound 1
For any shape
Complexity(S) ≤ n-1

For parallelogram-free shapes, we have
Complexity(S)=n-1

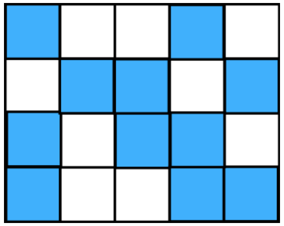


Parallelogram-free shape



Non Parallelogram-free shape

There exists 4 distinct points u,v,s,t in S with v-u=t-s



A shape *S* with *n* squares

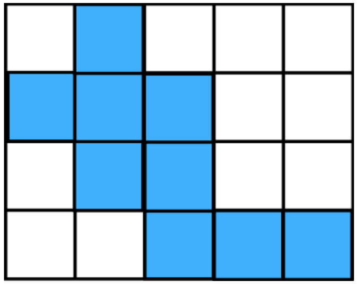
Bound 1

For any shape

Complexity(S) ≤ n-1

Bound 1
For any shape
Complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
complexity(S) =O(log(n))



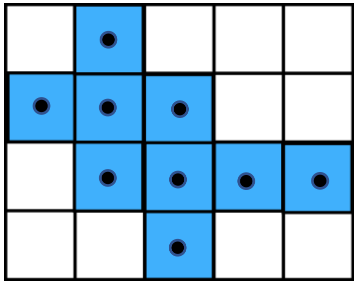
A HV-convex polyomino

It is a 4-connected,
horizontally and vertically convex shape

Bound 1
For any shape
Complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
complexity(S) =O(log(n))

Bound 3
For digital convex polyominoes
complexity(S) =O(log(log(n)))

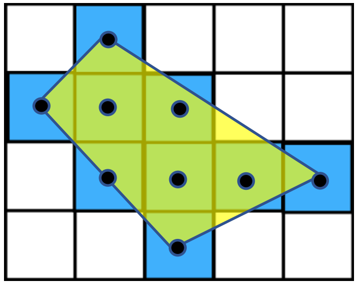


A digital convex shape

Bound 1
For any shape
Complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
complexity(S) =O(log(n))

Bound 3
For digital convex polyominoes
complexity(S) =O(log(log(n)))



A digital convex shape

Plan

I Problem Statement

II Shooting Algorithms

III Results

IV Proofs

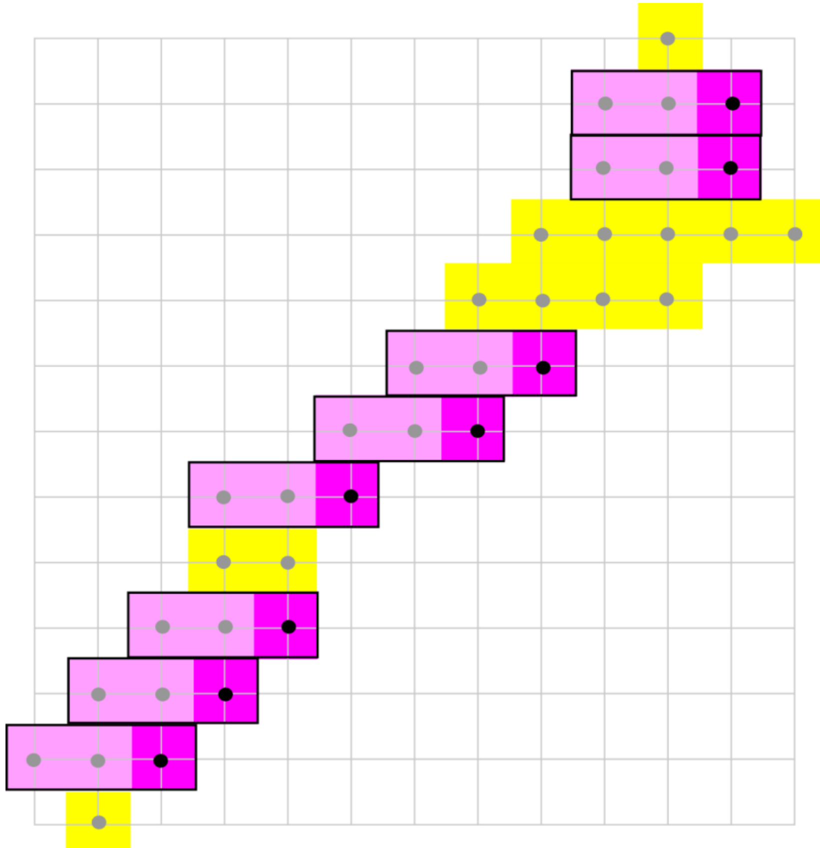
Plan

I Problem Statement

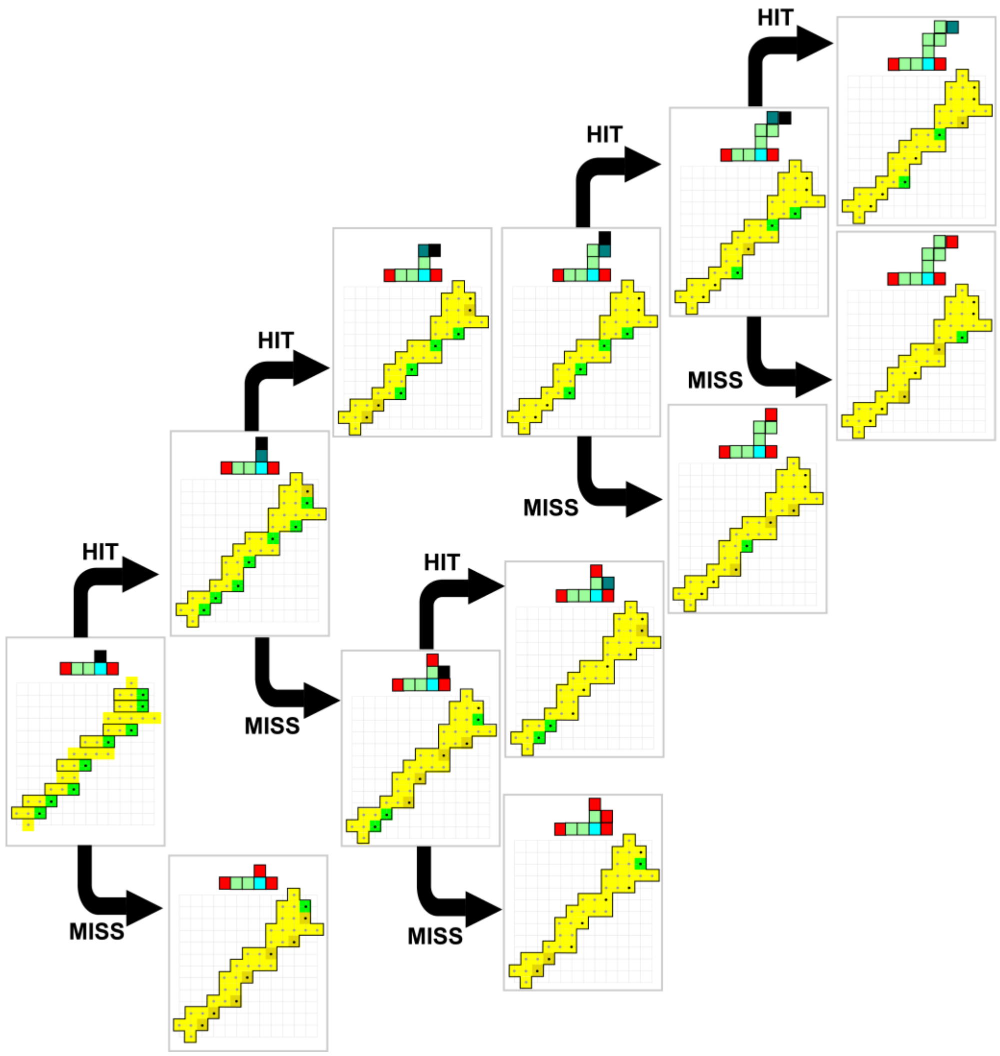
II Shooting Algorithms

III Results

IV Proofs



We use a property of monotonicity of HV-convex polyominoes…

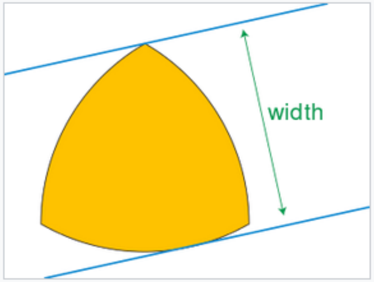


We build a staircase shooting algorithm with at most O(log(n)) misses

Blaschke–Lebesgue theorem

From Wikipedia, the free encyclopedia

In plane geometry the **Blaschke–Lebesgue theorem** states that the Reuleaux triangle has the least area of all curves of given constant width.[1] In the form that every curve of a given width has area at least as large as the Reuleaux triangle, it is also known as the **Blaschke–Lebesgue inequality**.[2] It is named after Wilhelm Blaschke and Henri Lebesgue, who published it separately in the early 20th century.

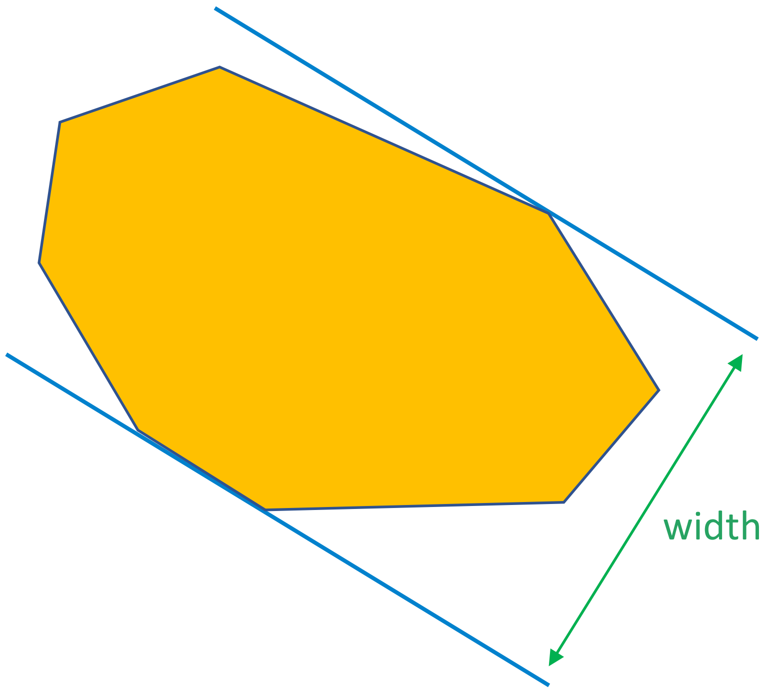


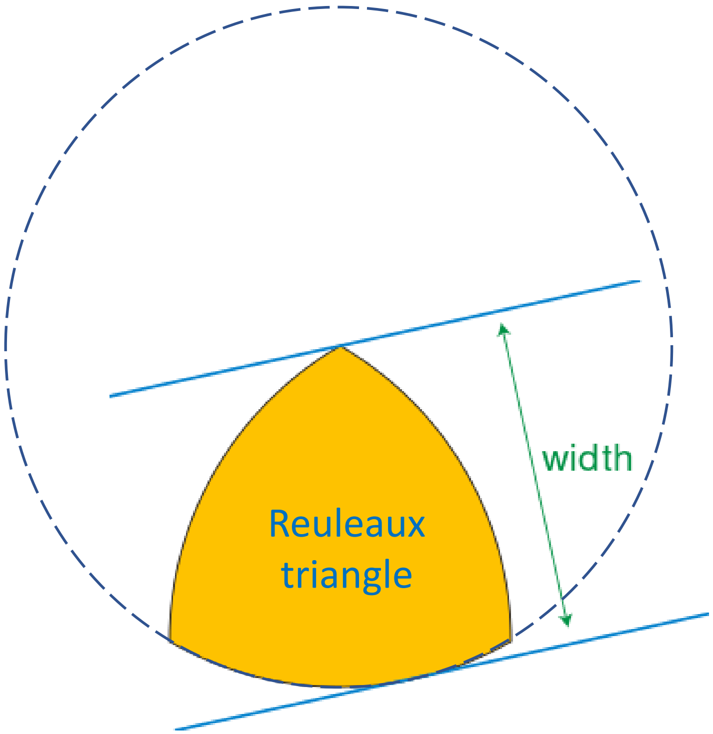
A Reuleaux triangle, a curve of constant width whose area is minimum among all convex sets with the same width

Contents [hide]

Statement [edit]

The width of a convex set K in the Euclidean plane is defined as the minimum distance between any two parallel lines that enclose it. The two minimum-distance lines are both necessarily tangent lines to K, on opposite sides. A curve of constant width is the boundary of a convex set with the property that, for every direction of parallel lines, the two tangent lines with that direction that are tangent to opposite sides of the curve are at a distance equal to the width. These curves include both the circle and the Reuleaux triangle, a curved triangle formed from arcs of three equal-radius



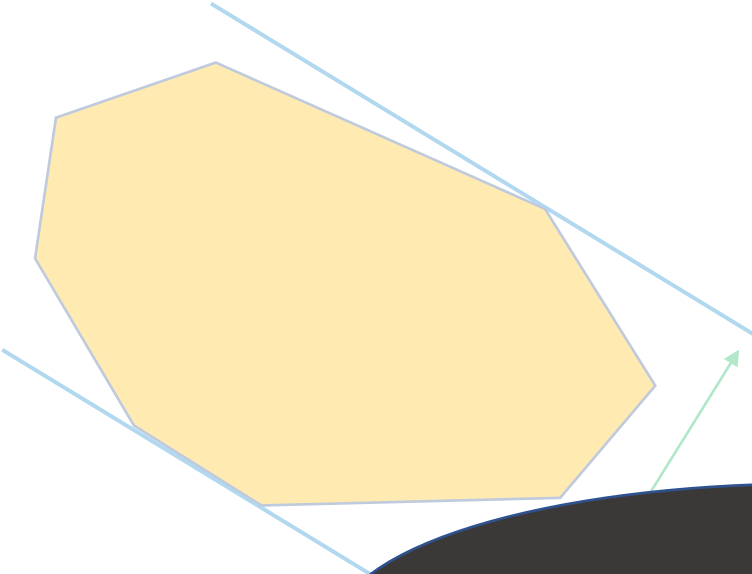


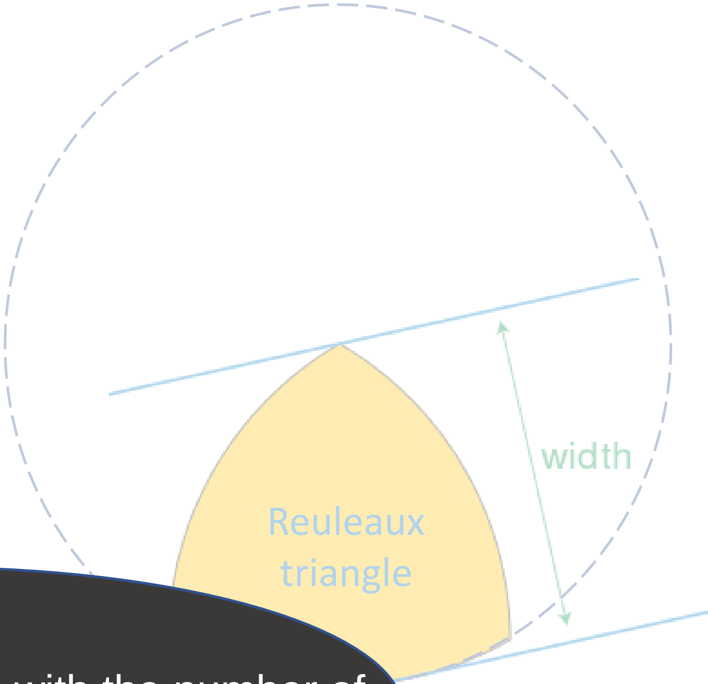
Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by

$$area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$$

(Equality is achieved for Reuleaux triangles).





Is there a discrete version with the number of points instead of the area ?

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by

$$area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$$

(Equality is achieved for Reuleaux triangles).

Barany-Füredi Inequality *(2001)*

The area of a discrete shape S is bounded by $width \leq \lfloor 4/3\ diam \rfloor + 1$
where $diam$ is the maximum number of points of S on a line
and $width$ is the arithmetic width (tight bound).

Is there a discrete version with the number of points instead of the area ?

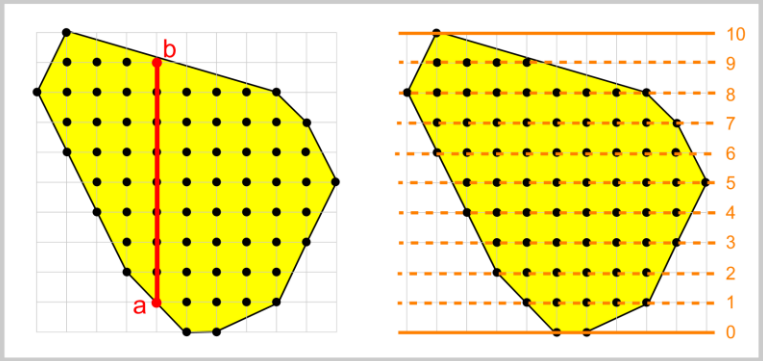
Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$

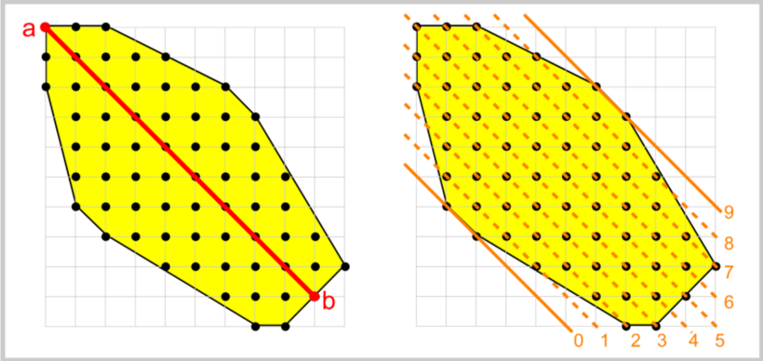
(Equality is achieved for Reuleaux triangles).

Barany-Füredi Inequality *(2001)*

The area of a discrete shape *S* is bounded by $width \leq \lfloor 4/3\, diam \rfloor + 1$
where $diam$ is the maximum number of points of *S* on a line
and $width$ is the arithmetic width (tight bound).



$diam = 9$ $width = 10$



$diam = 10$ $width = 9$

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$
(Equality is achieved for Reuleaux triangles).

Barany-Füredi Inequality *(2001)*

The area of a discrete shape S is bounded by $width \leq \lfloor 4/3 \; diam \rfloor + 1$
where $diam$ is the maximum number of points of S on a line
and $width$ is the arithmetic width (tight bound).

Pick formula & …

***Discrete* Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)**

The number n of a digital convex shape S is bounded by $n \geq \frac{1}{4} width^2 + 2$

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$
(Equality is achieved for Reuleaux triangles).

Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number n of a digital convex shape S is bounded by $n \geq \frac{1}{4} width^2 + 2$

(not tight)

Blaschke-Lebesgue Inequality *(1914)*

The area of a convex shape is bounded by $area \geq \frac{1}{2}\left(\pi - \sqrt{3}\right) width^2$

(Equality is achieved for Reuleaux triangles).

Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number n of a digital convex shape S is bounded by $n \geq \frac{1}{4} width^2 + 2$

(not tight)

Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number n of a digital convex shape *S* is bounded by $n \geq \frac{1}{4} width^2 + 2$

(not tight)

Node

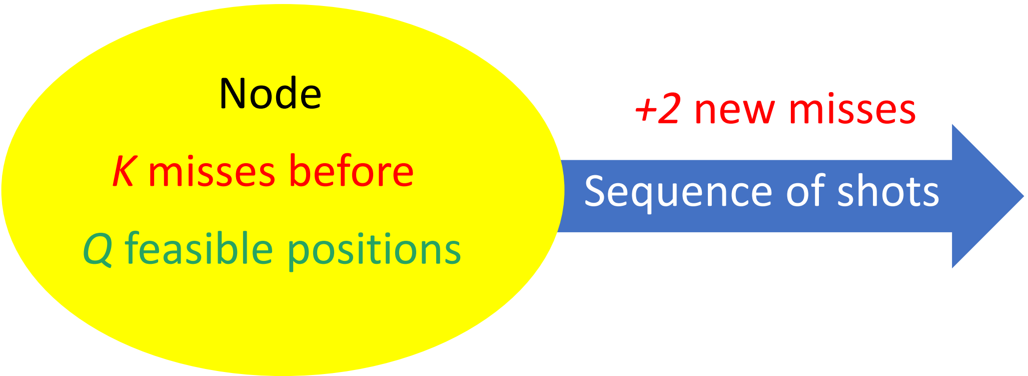
K misses before

Q feasible positions

Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number *n* of a digital convex shape *S* is bounded by $n \geq \frac{1}{4} width^2 + 2$

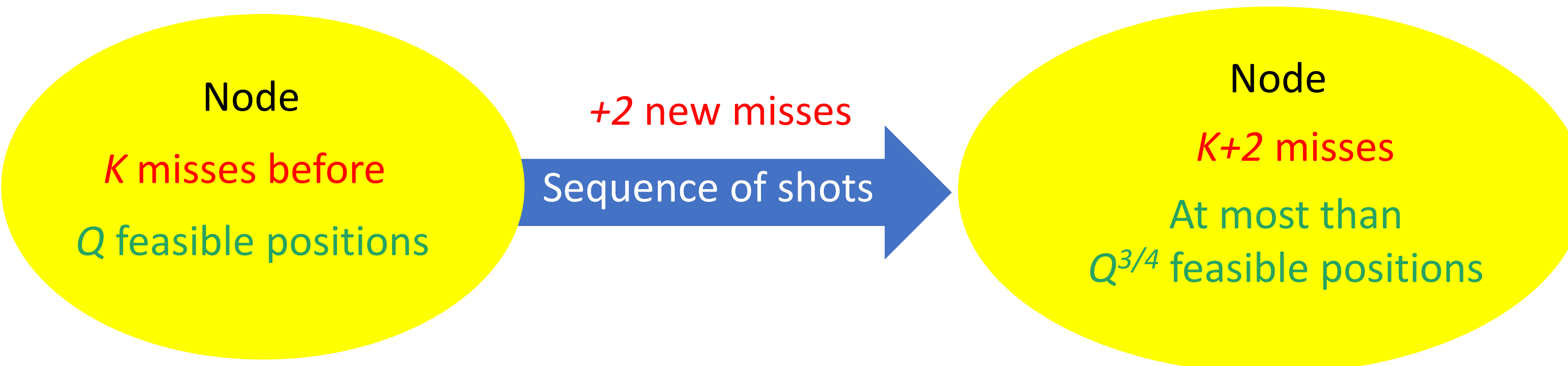
(not tight)



Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number *n* of a digital convex shape *S* is bounded by $n \geq \frac{1}{4} width^2 + 2$

(not tight)



Discrete Blaschke-Lebesgue Inequality (L. Crombez, G. Da Fonseca, Y.G)

The number *n* of a digital convex shape *S* is bounded by $n \geq \frac{1}{4} width^2 + 2$

(not tight)

Conclusion

Three bounds on the Battleship complexity of a shape….

Bound 1
For any shape
Complexity(S) ≤ n-1

Bound 2
For HV-convex polyominoes
complexity(S) =O(log(n))

Bound 3
For digital convex polyominoes
complexity(S) =O(log(log(n)))

Conclusion

Three bounds on the Battleship complexity of a shape….

Bound 1
For any shape
Complexity(S) $\leq$ n-1

Bound 2
For HV-convex polyominoes
complexity(S) =O(log(n))

Bound 3
For digital convex polyominoes
complexity(S) =O(log(log(n)))

A lot of open questions…

Conclusion

Three bounds on the Battleship complexity of a shape….

Bound 1
For any shape
Complexity(S) ≤ n-1

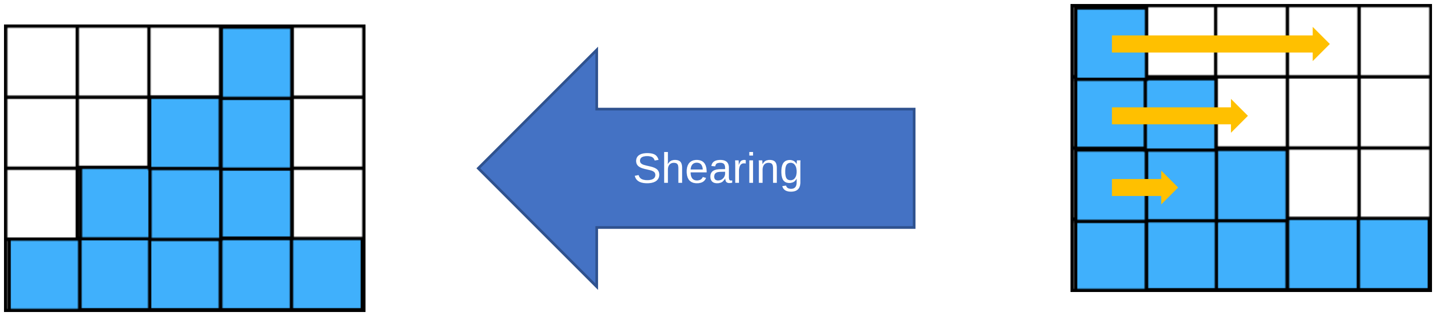
Bound 2
For HV-convex polyominoes
complexity(S) =O(log(n))

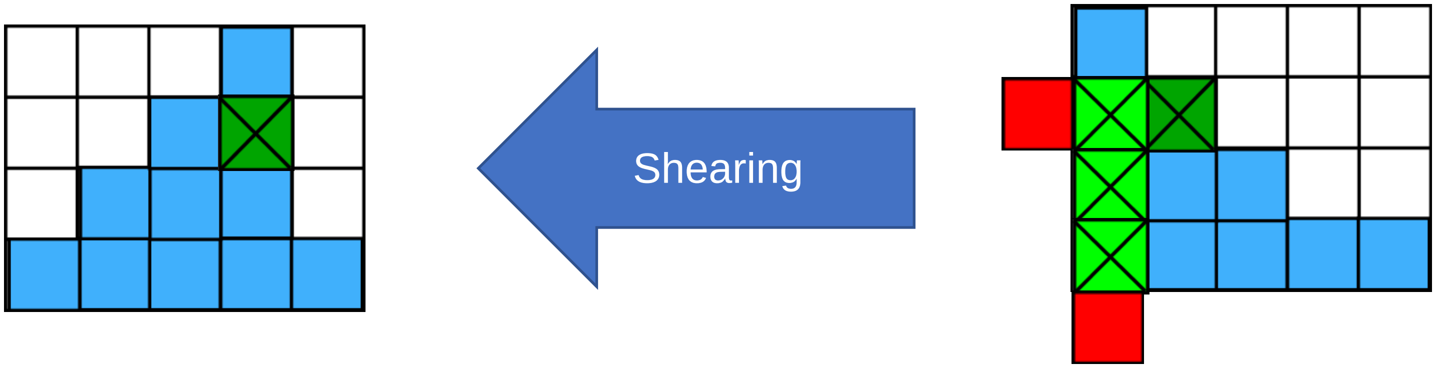
Bound 3
For digital convex polyominoes
complexity(S) =O(log(log(n)))

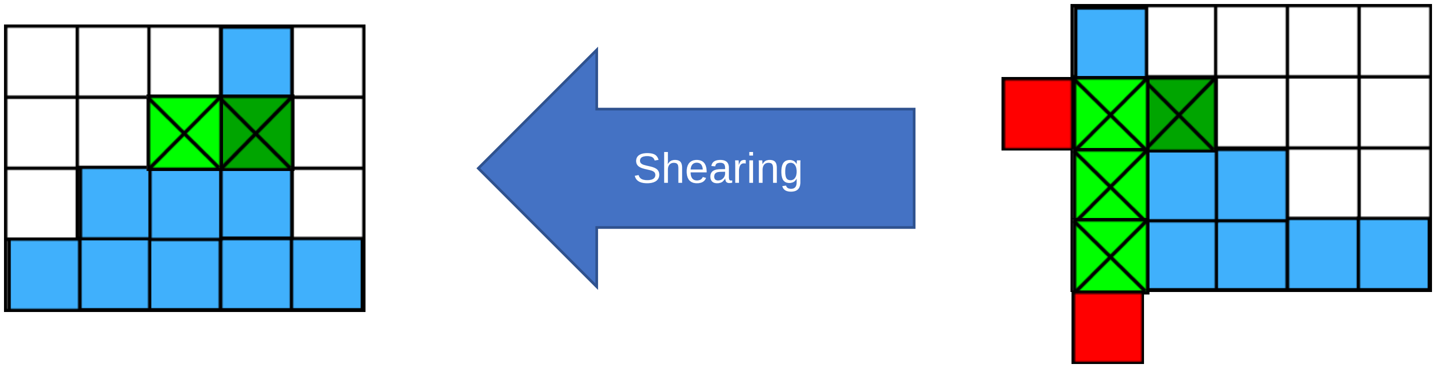
A lot of open questions…

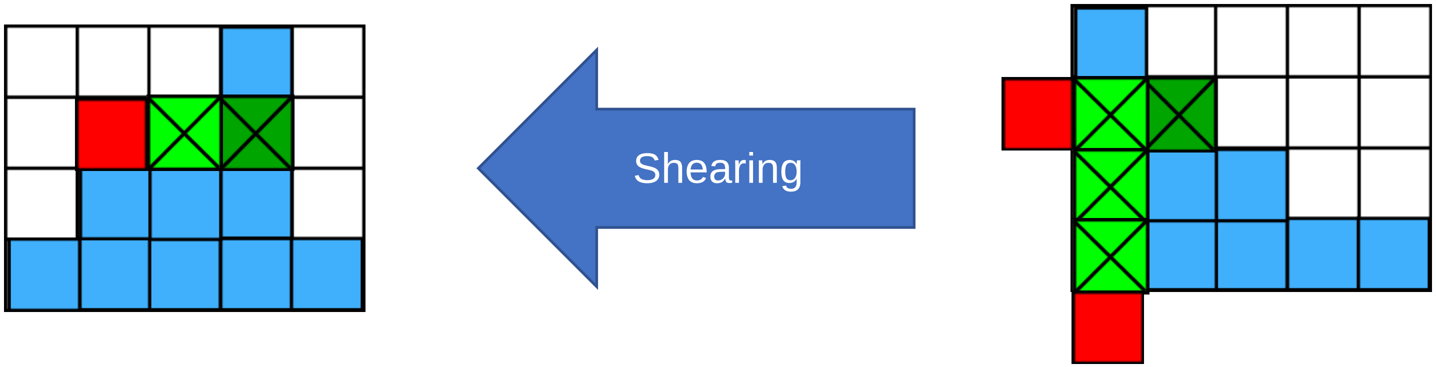
Complexity(A+B) ≤ Complexity(A)+Complexity(B) ?

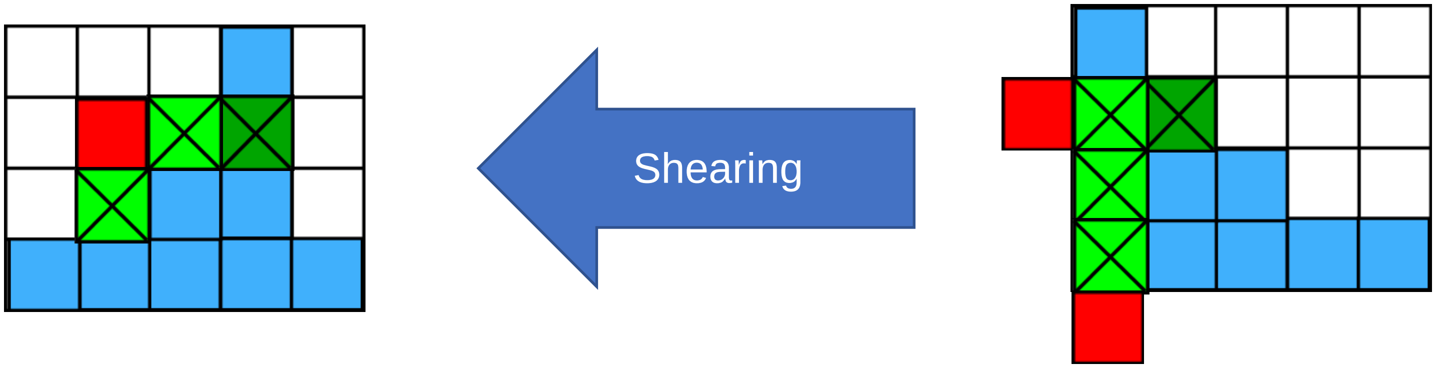
Thanks a lot for your attention…

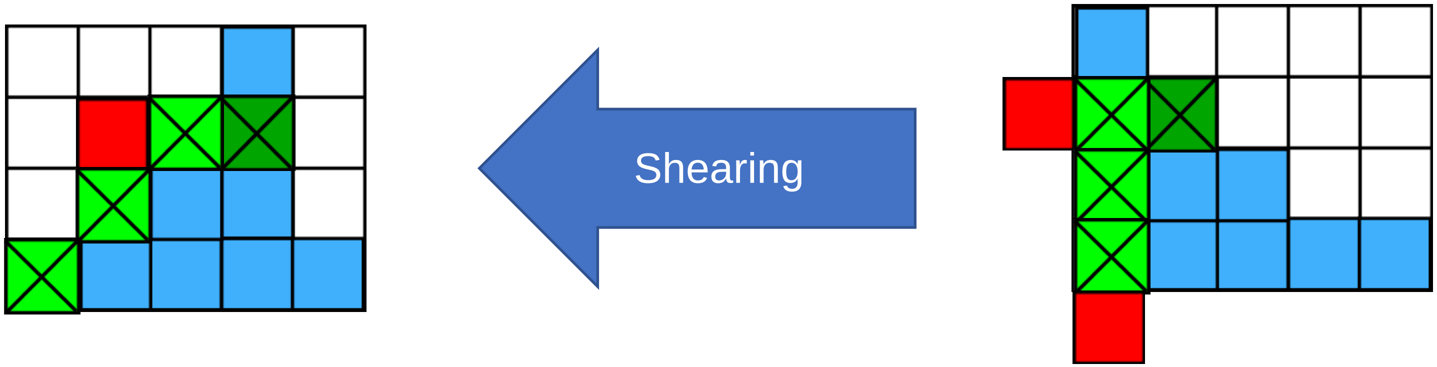

Shearing

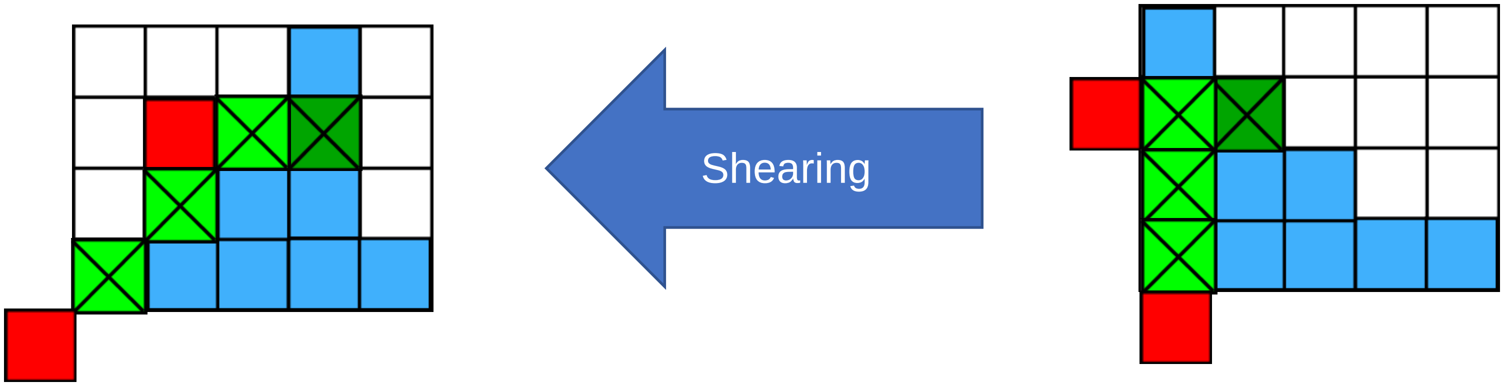

Shearing


Shearing


Shearing


Shearing


Shearing



The Battleship complexity is invariant by isomorphisms…