Hypergraphs and the Maker-Breaker game: a structural approach

PhD defense of Florian Galliot

3rd July 2023

Supervisors: Sylvain Gravier and Isabelle Sivignon

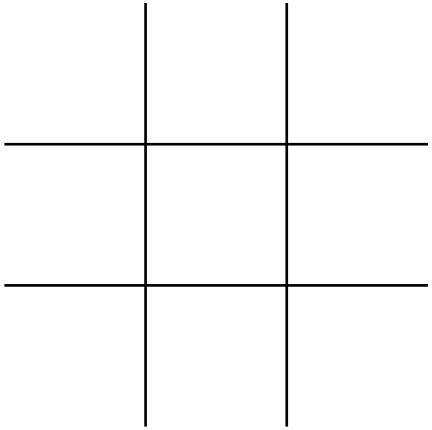
The Maker-Breaker game on hypergraphs of rank 3

1. Introduction
2. Structural result
3. Algorithmic result
4. Conclusion

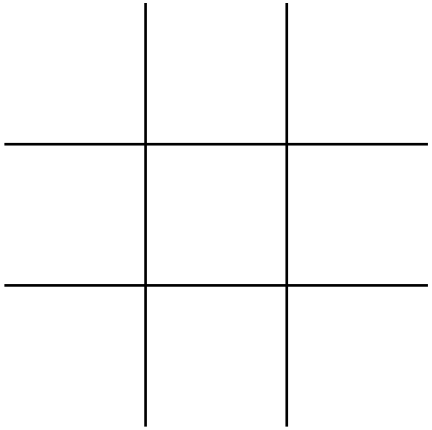
The Maker-Breaker game on hypergraphs of rank 3

1. Introduction
2. Structural result
3. Algorithmic result
4. Conclusion

Example 1: (asymmetrical) tic-tac-toe

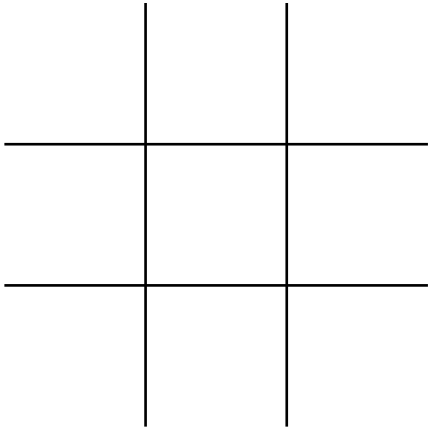


Example 1: (asymmetrical) tic-tac-toe



Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

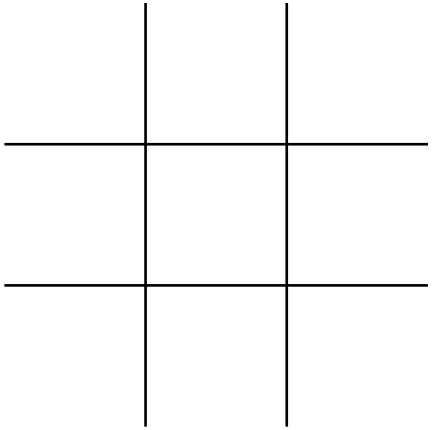
Example 1: (asymmetrical) tic-tac-toe



Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

Example 1: (asymmetrical) tic-tac-toe

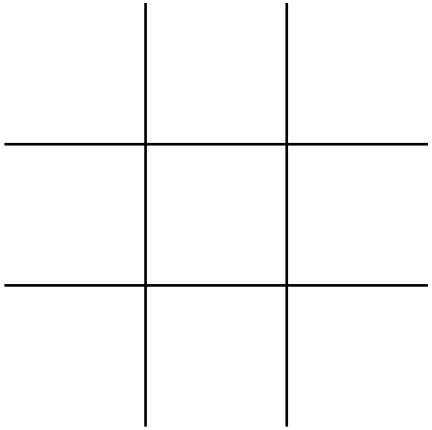


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her.

Example 1: (asymmetrical) tic-tac-toe

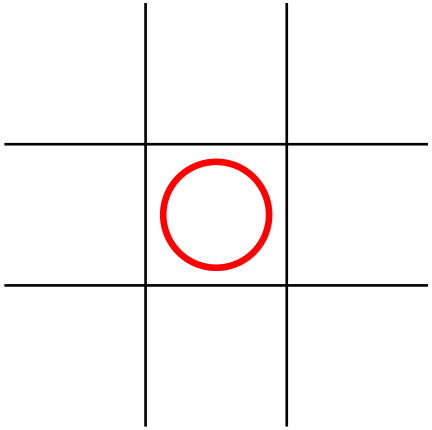


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

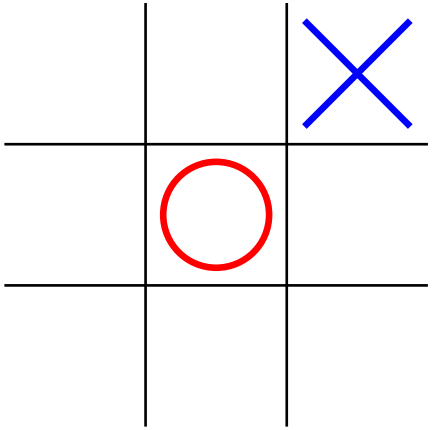


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

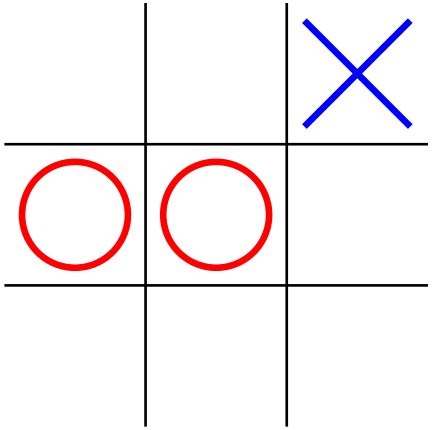


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

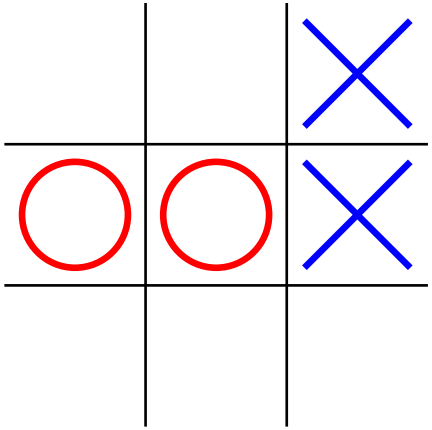


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

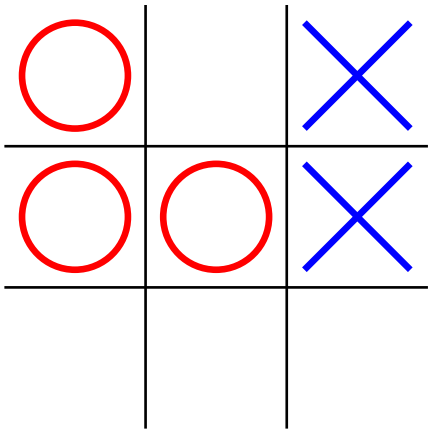


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

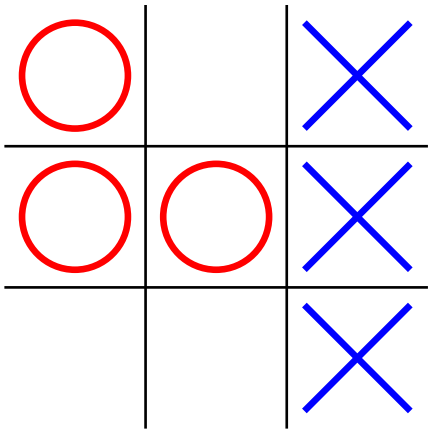


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

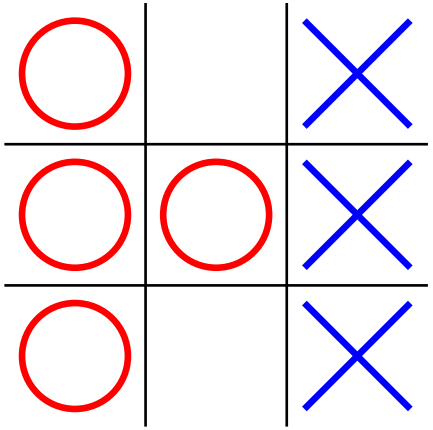


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

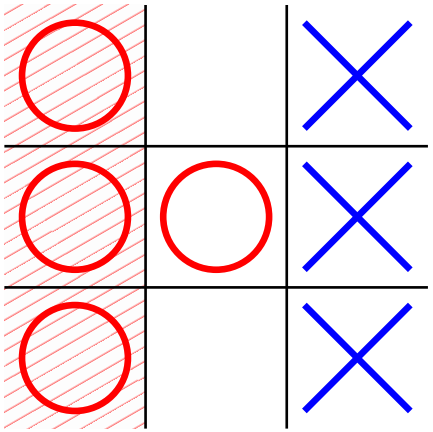


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe

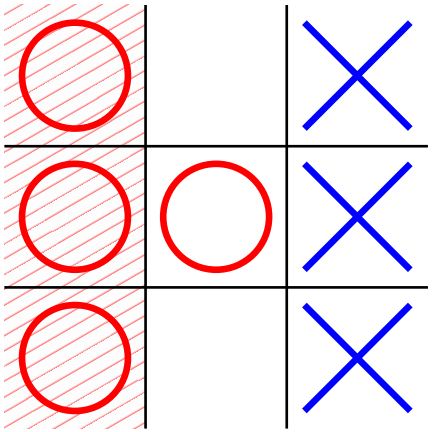


Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

Example 1: (asymmetrical) tic-tac-toe



Tic-tac-toe is a *Maker-Maker* game: both players aim at **making** an alignment of three symbols.

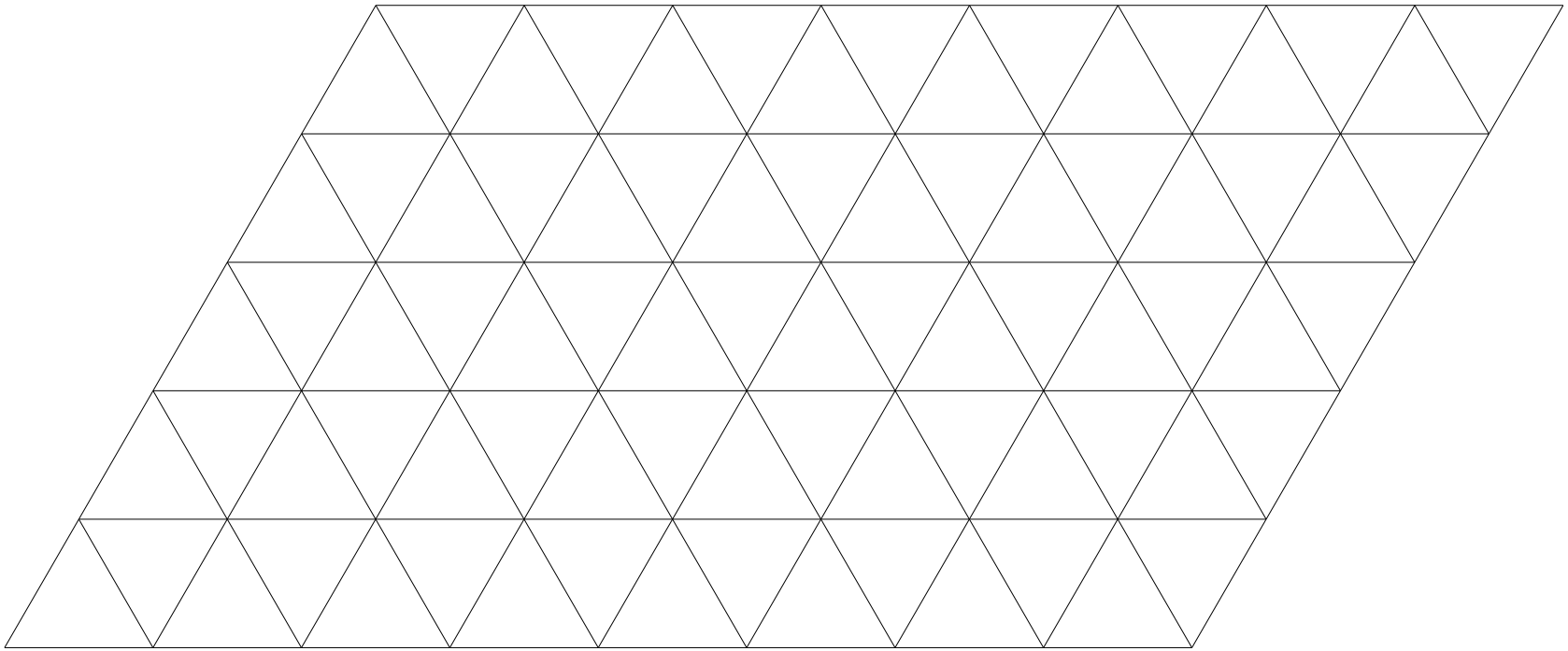
- Outcome: draw.

We can turn it into a *Maker-Breaker* game: Alice (Maker) plays first and aims at aligning three of her symbols; Bob (Breaker) aims at **blocking** her. (Remark: no draw is possible!)

- Outcome: Maker wins.

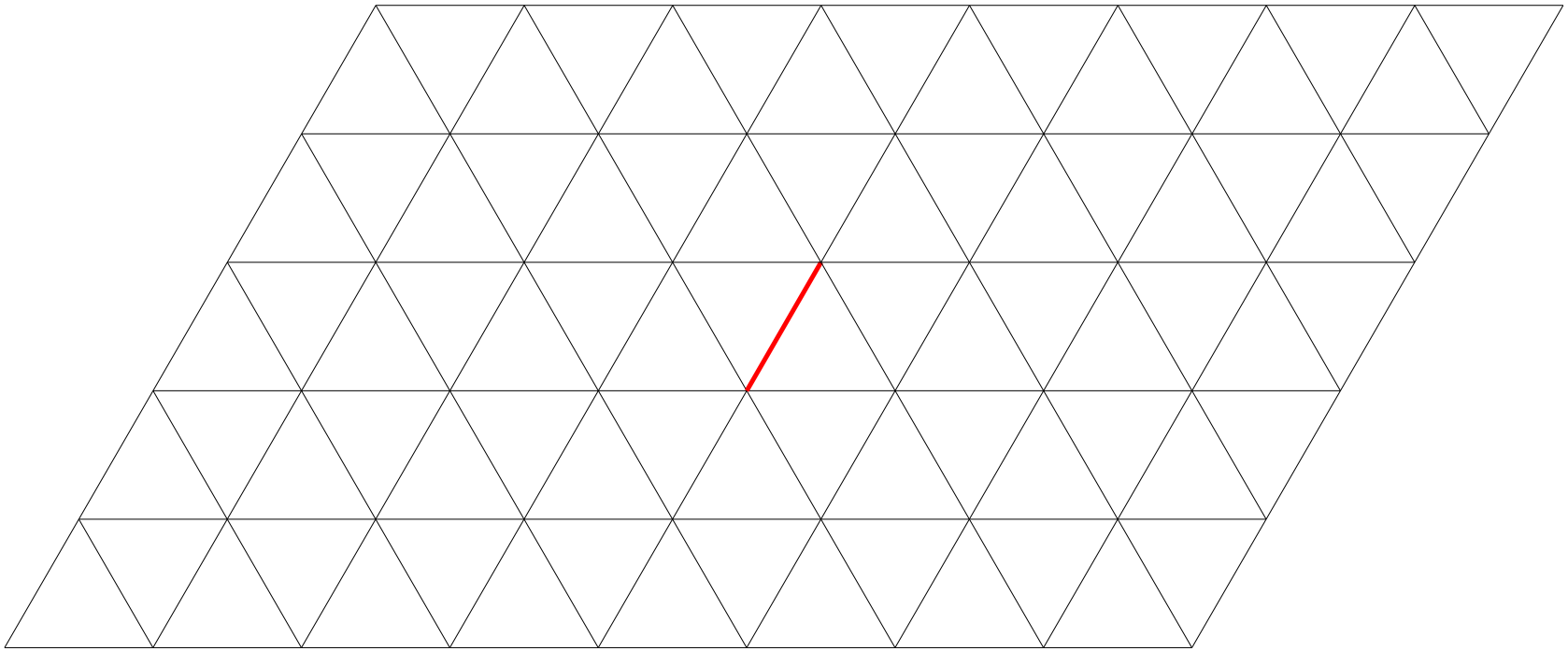
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



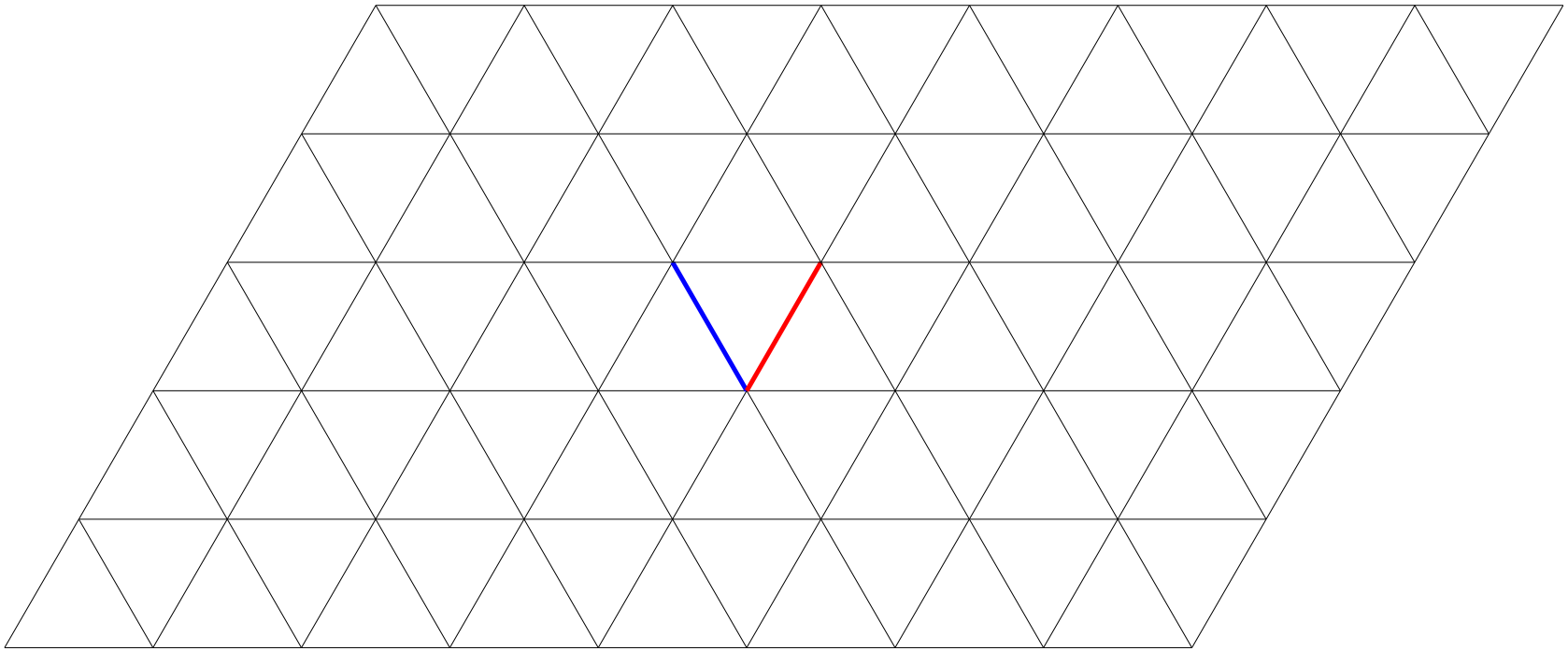
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



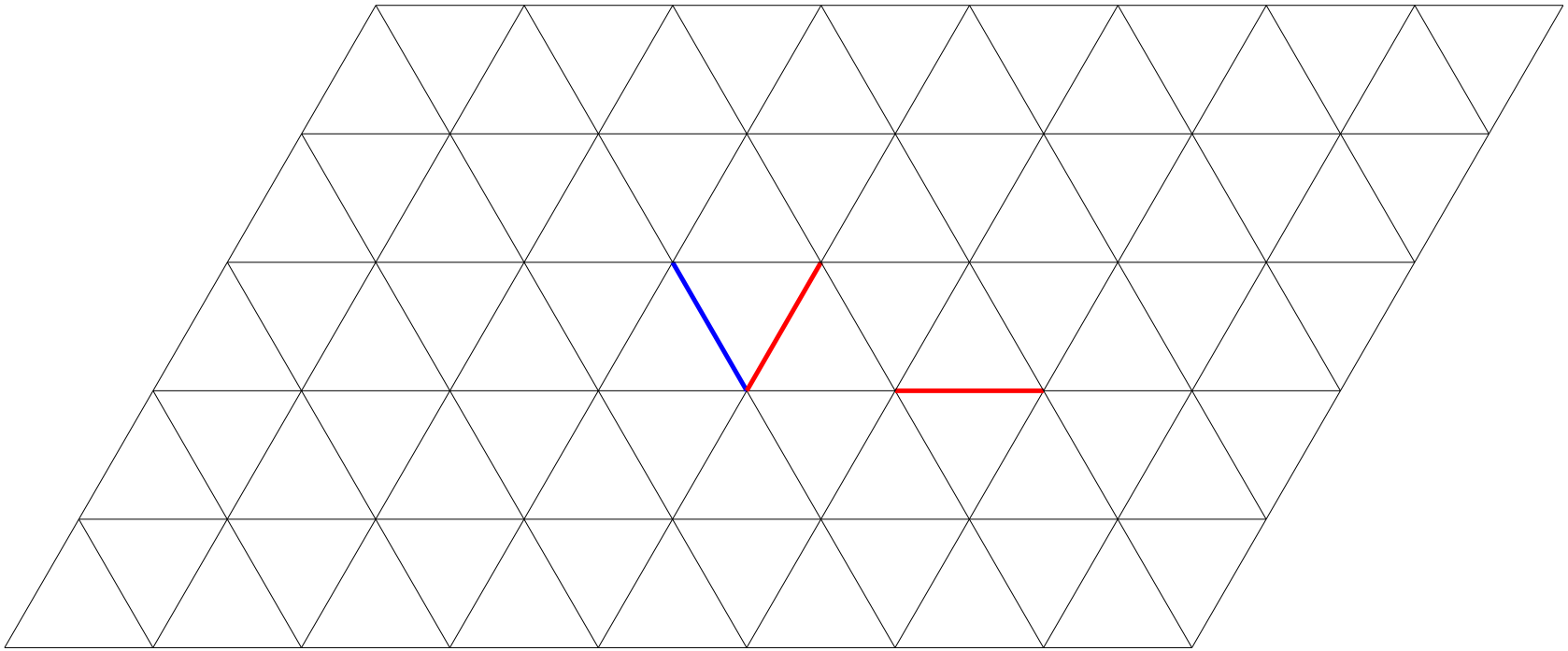
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



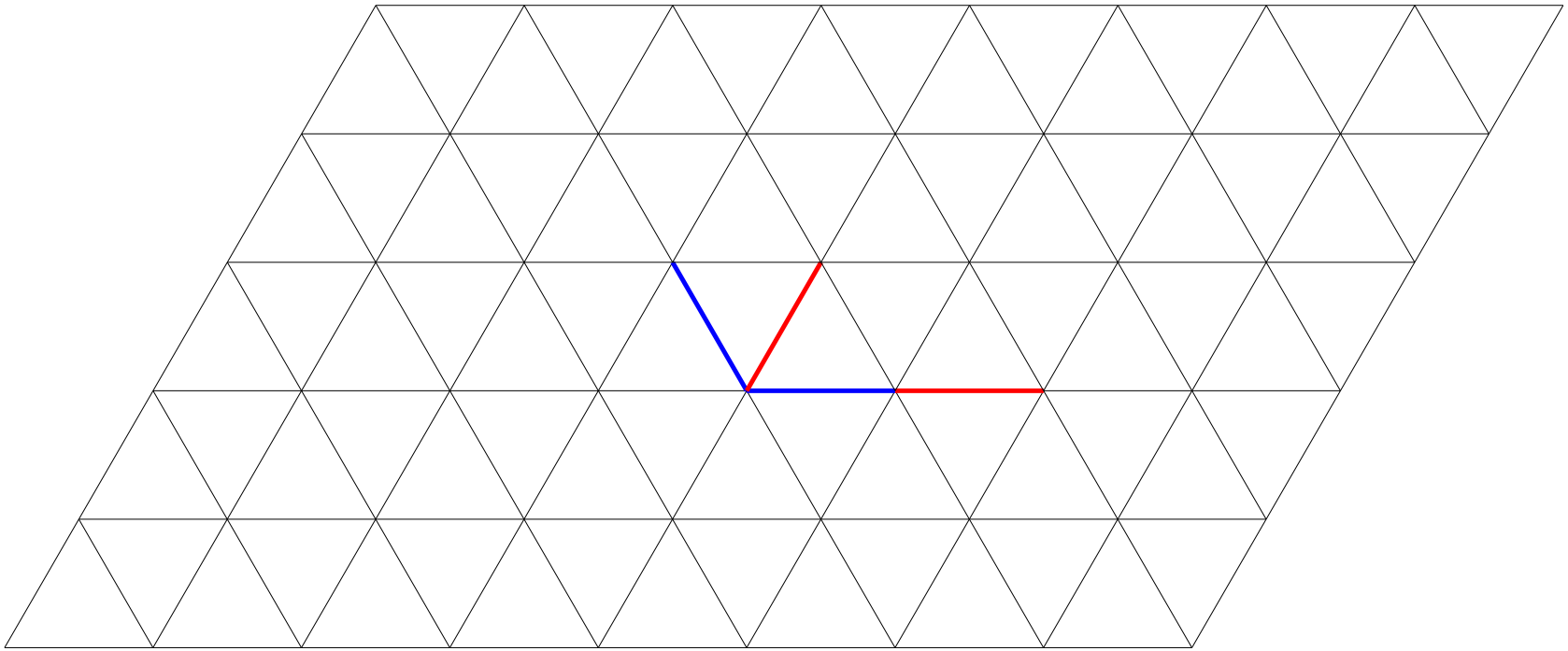
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



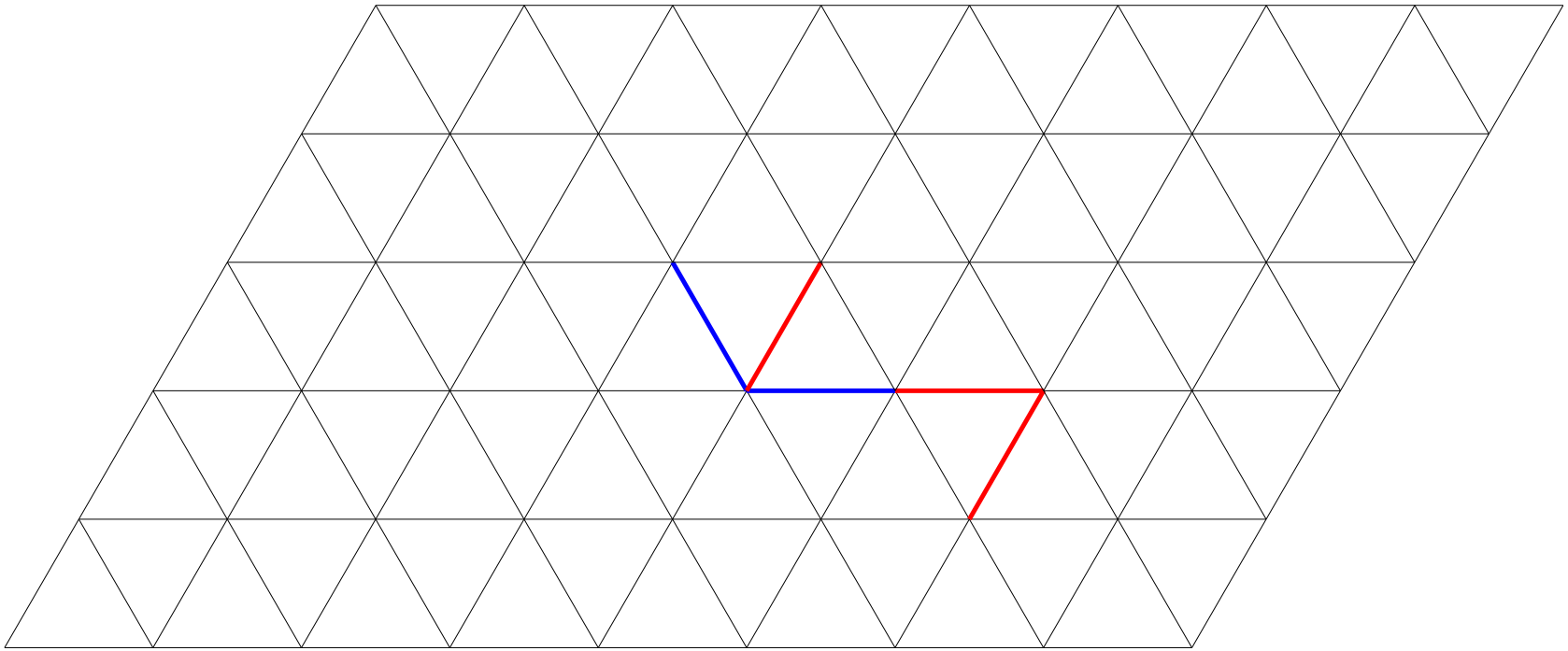
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



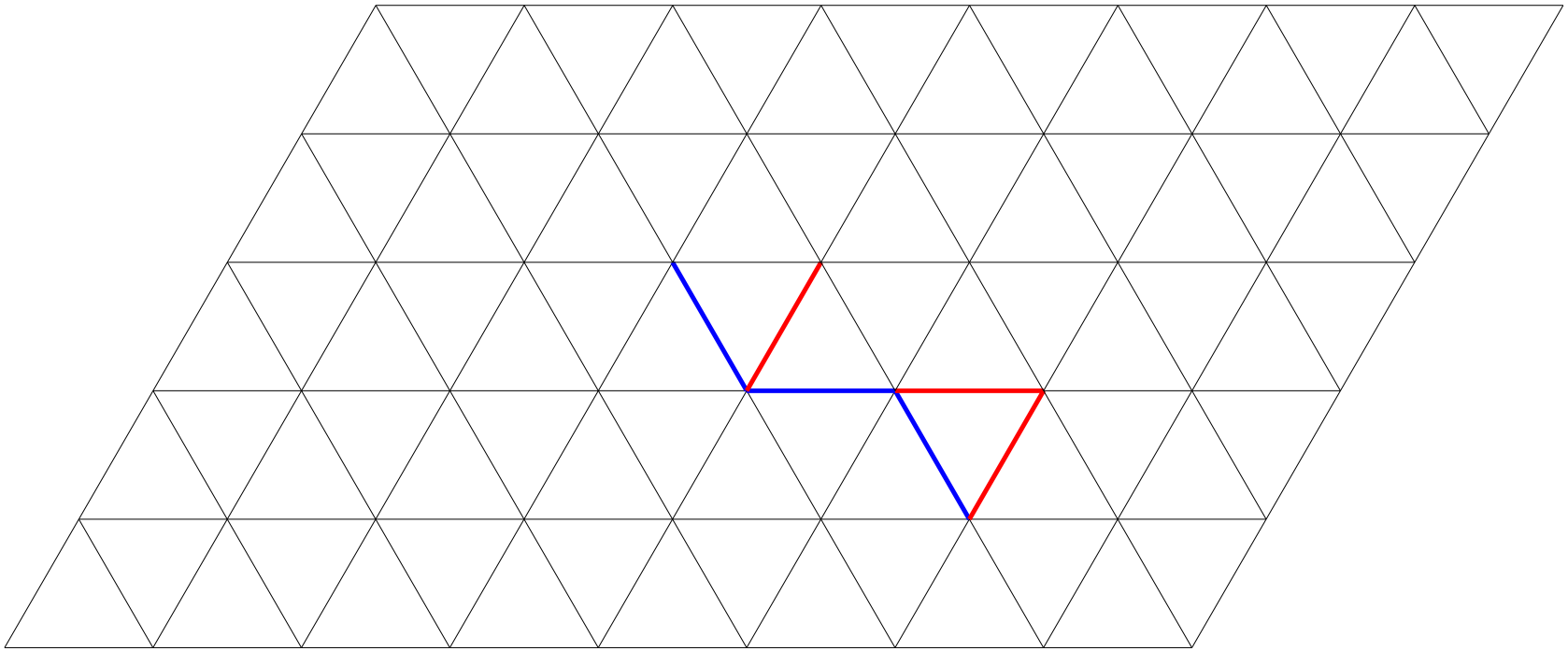
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



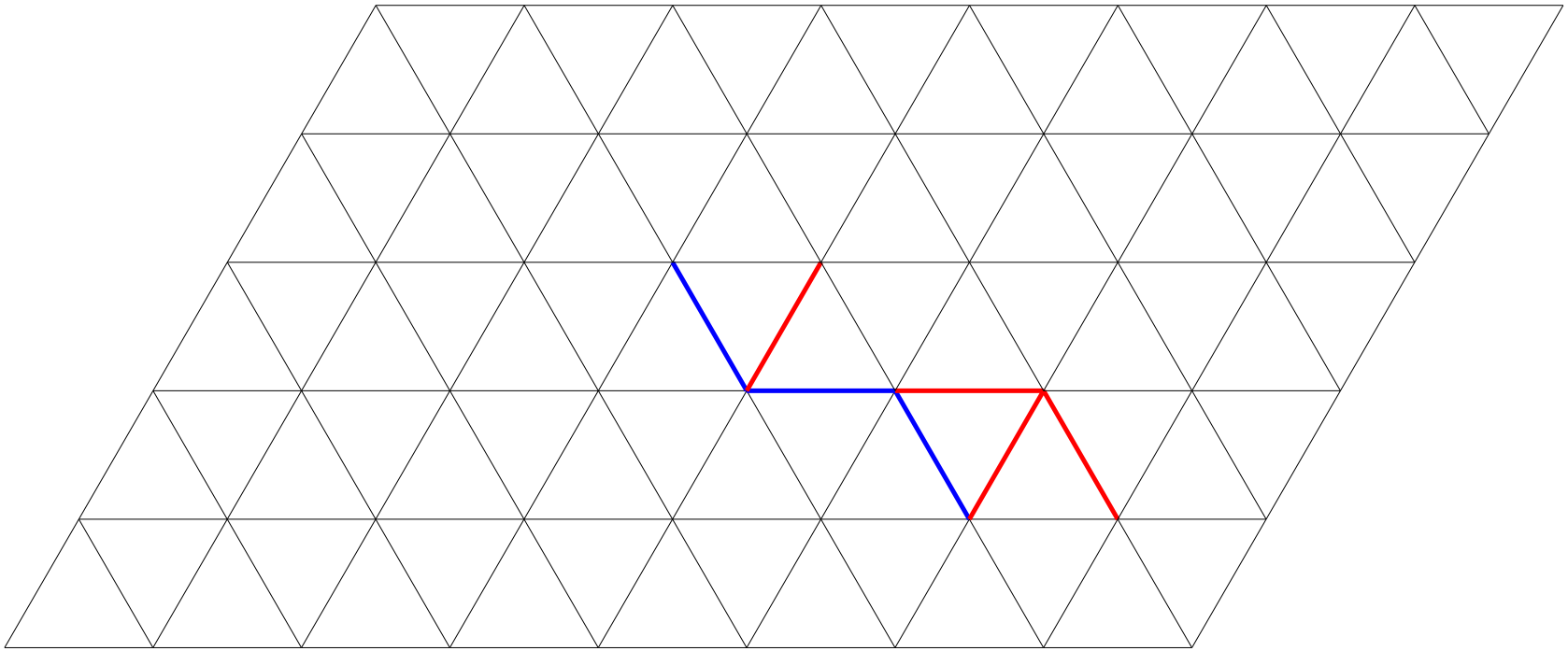
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



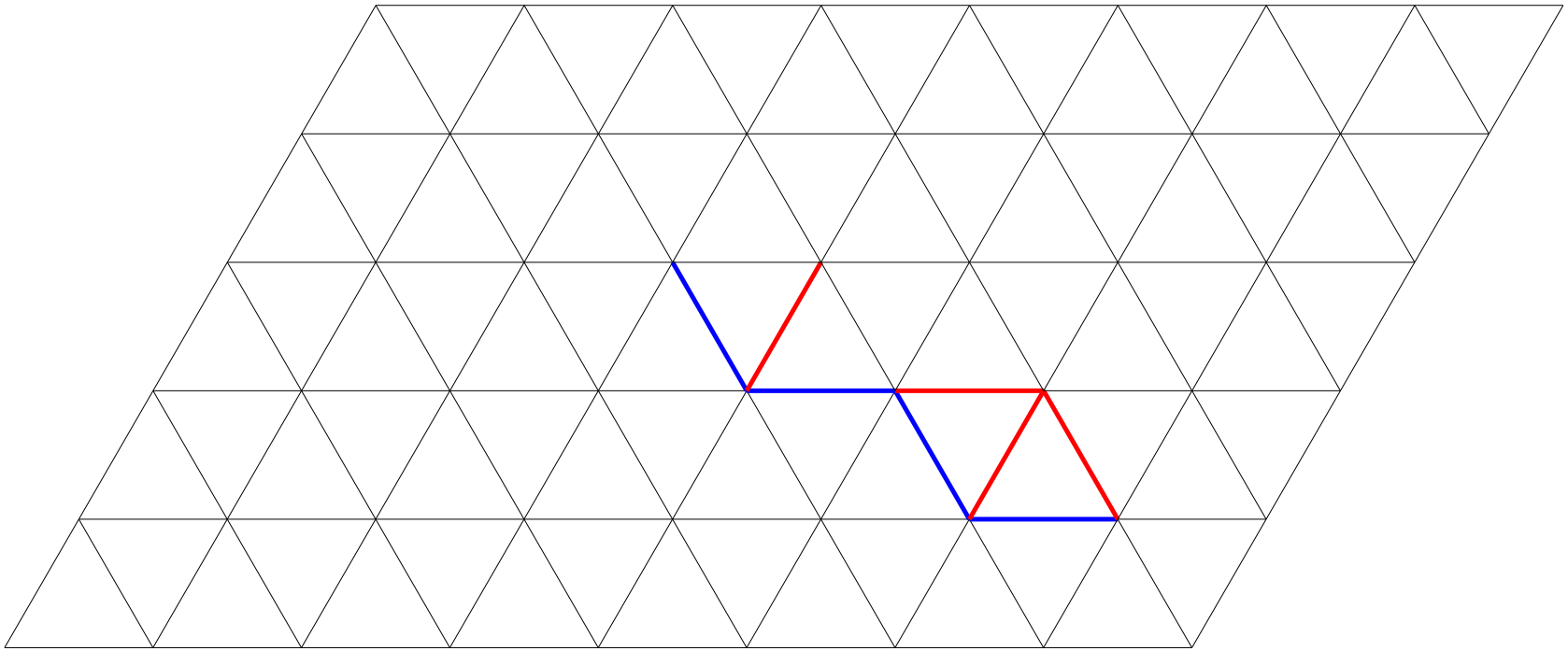
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



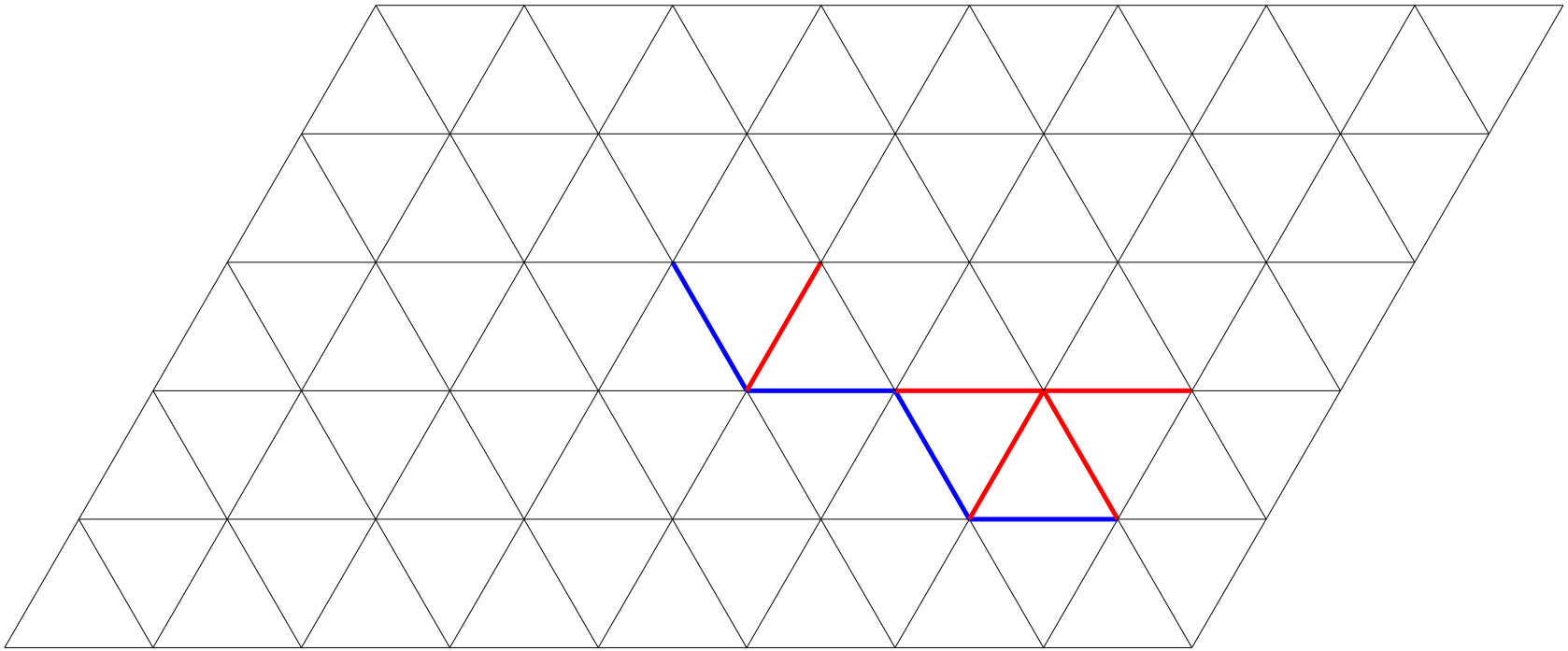
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



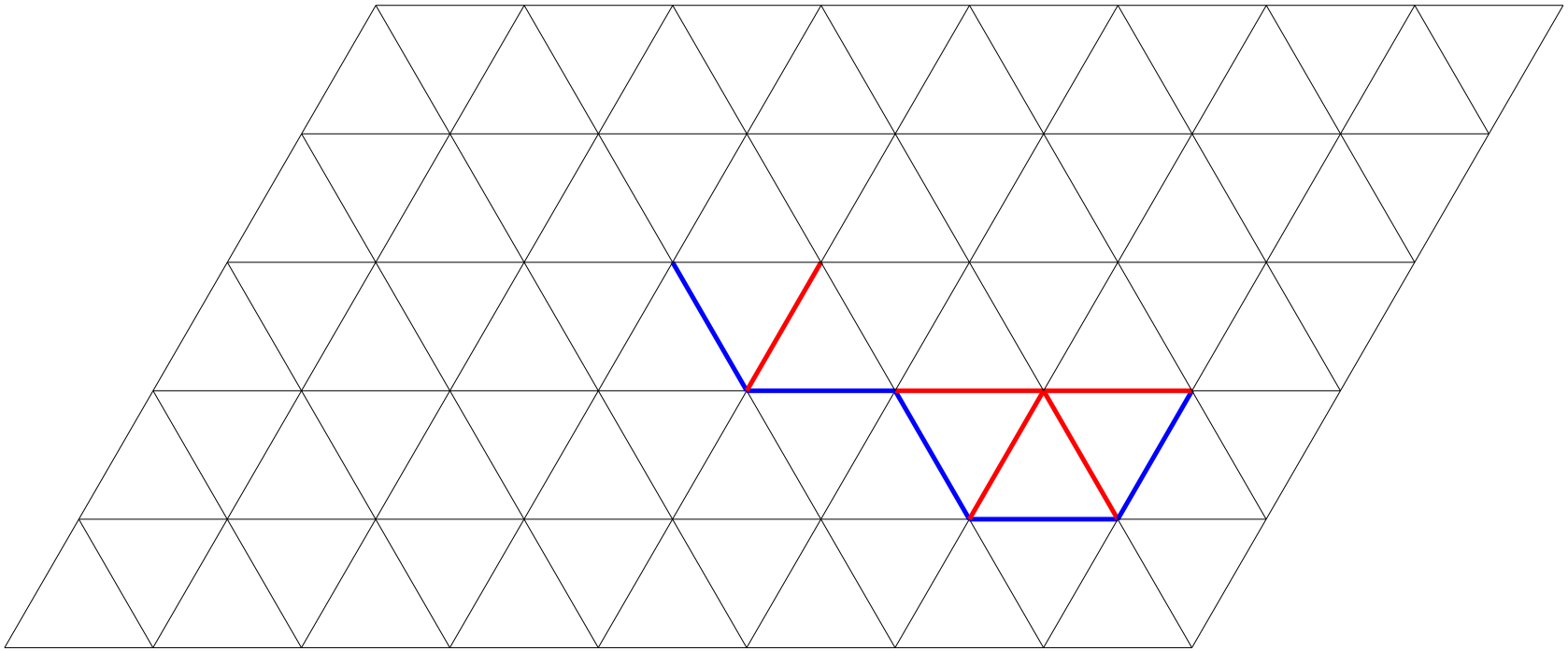
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



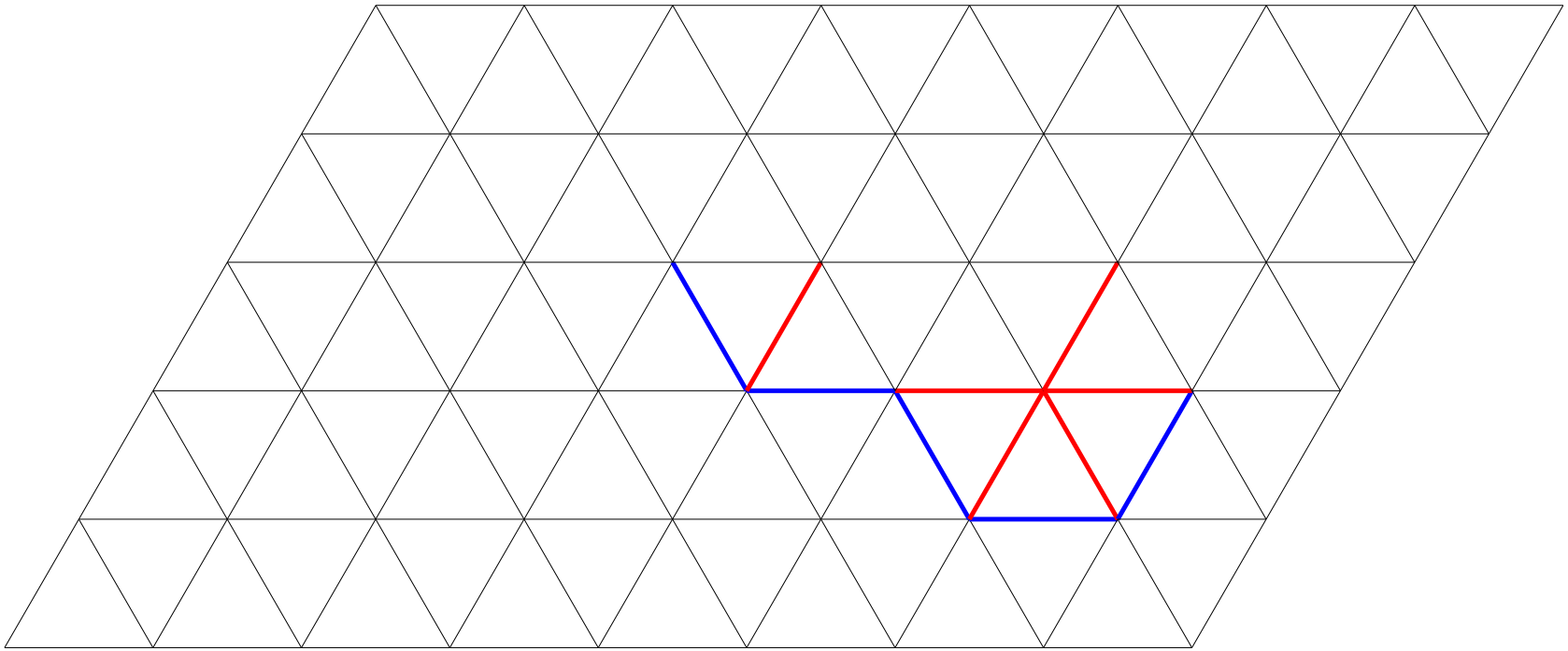
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



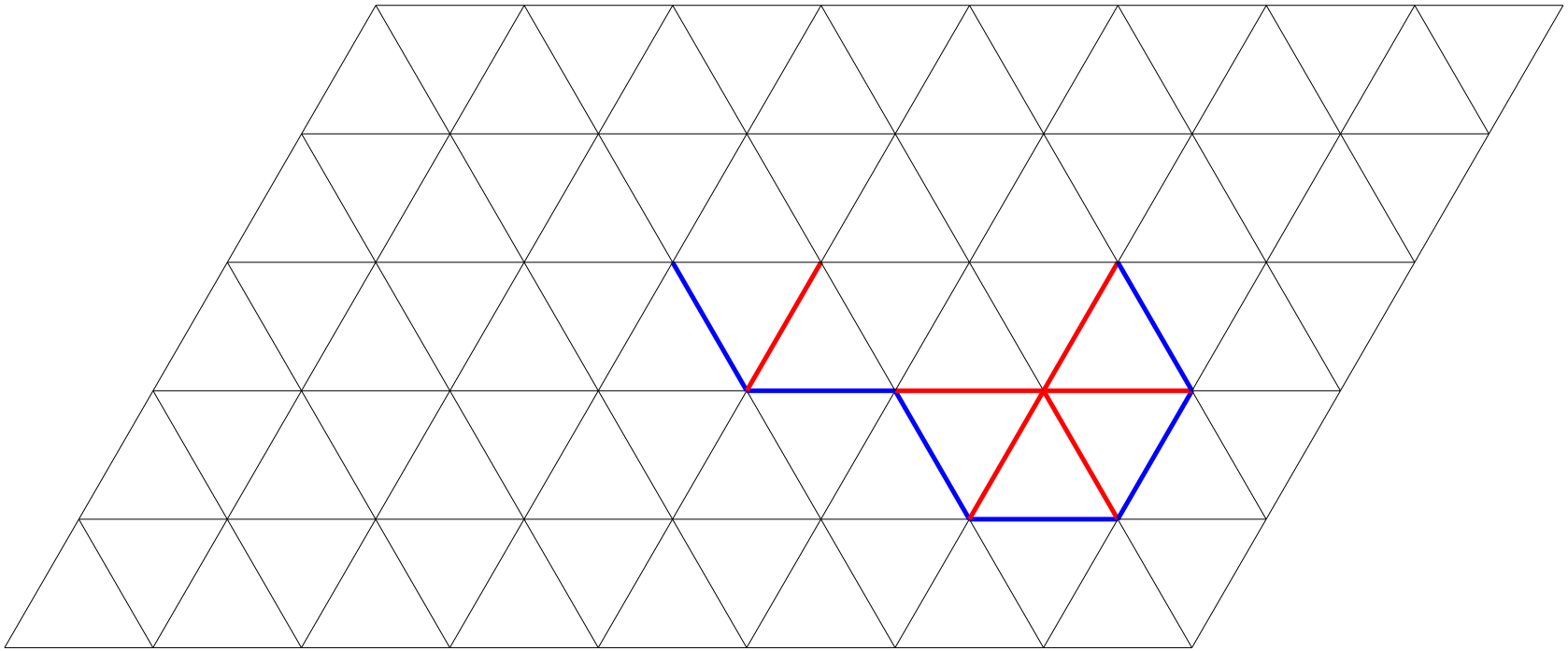
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



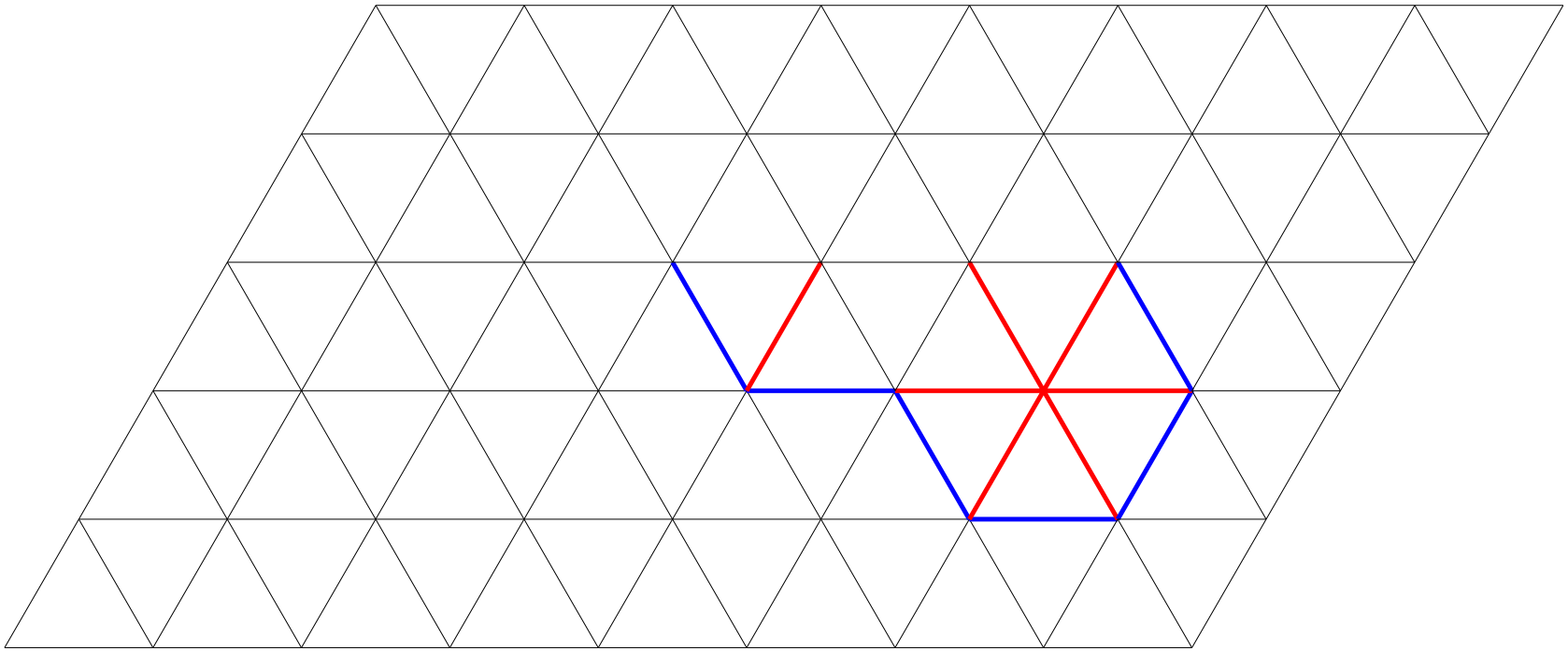
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



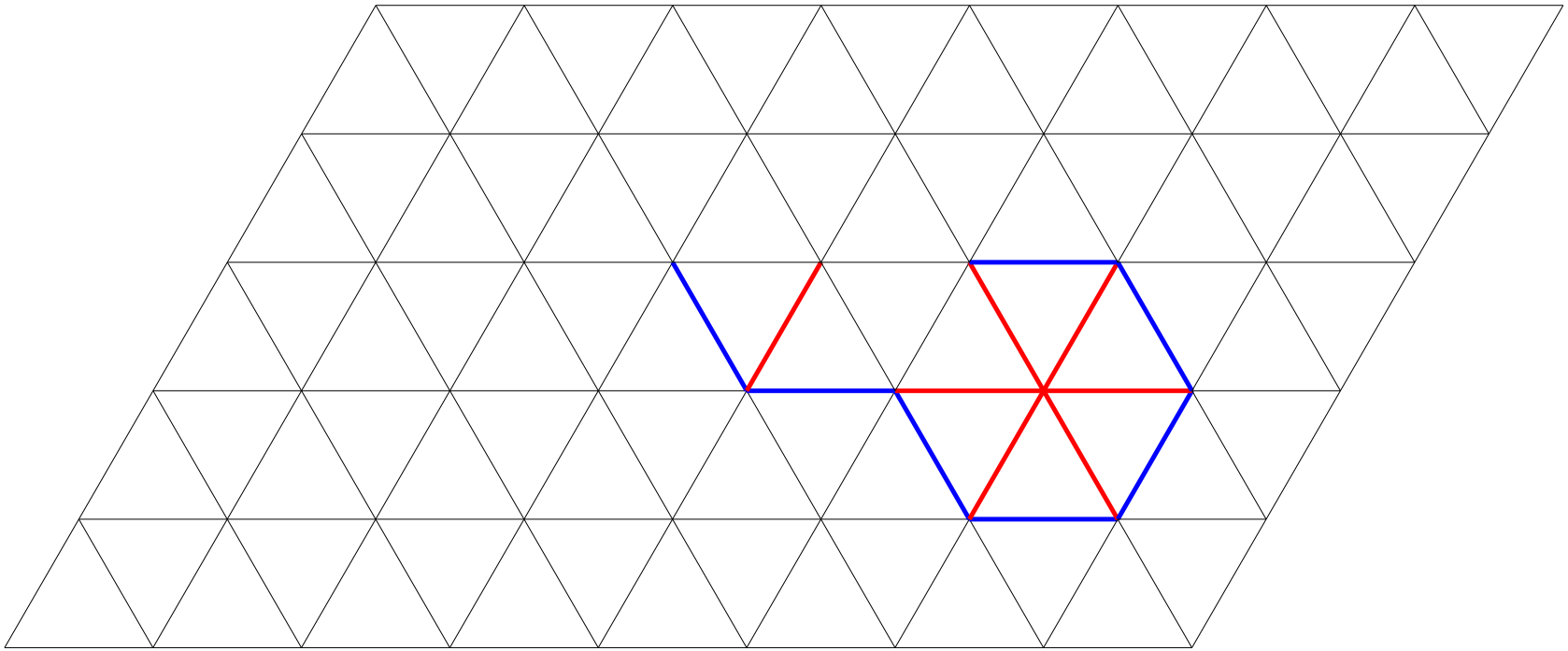
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



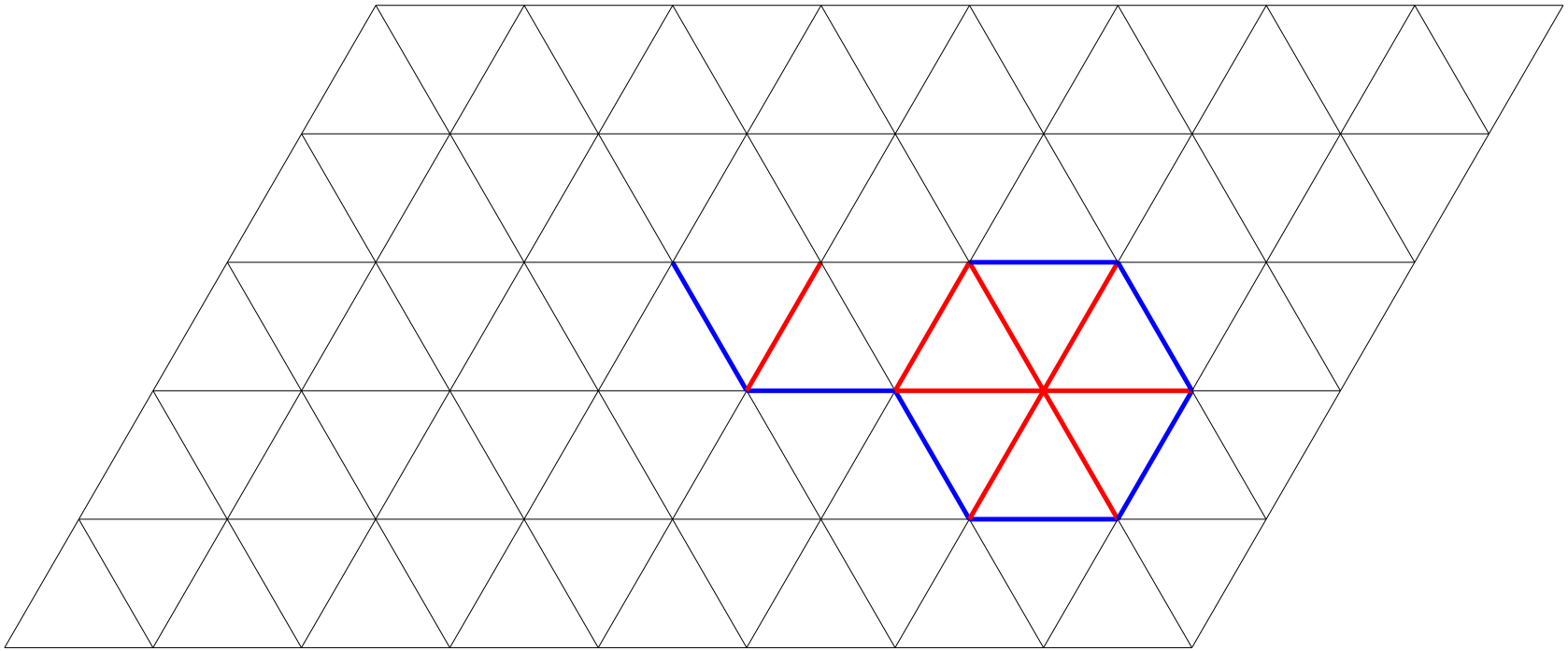
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



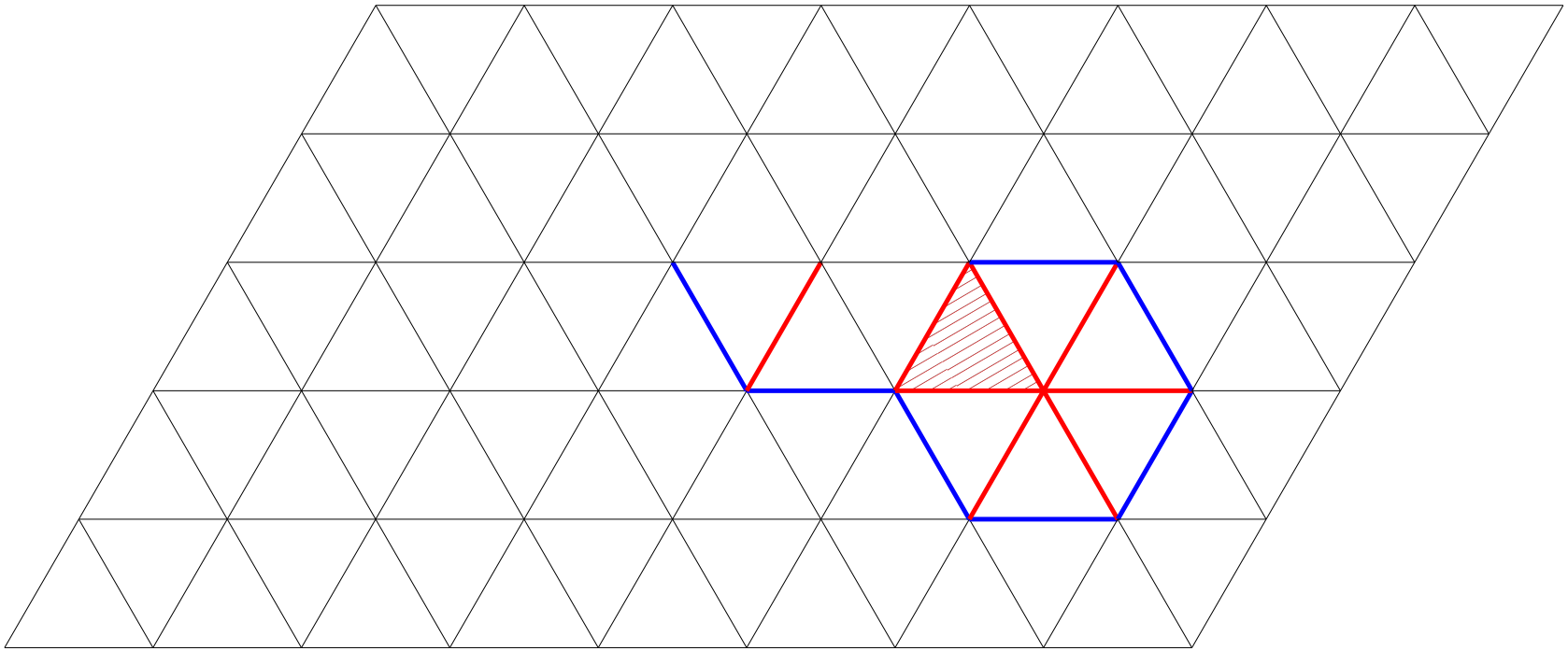
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



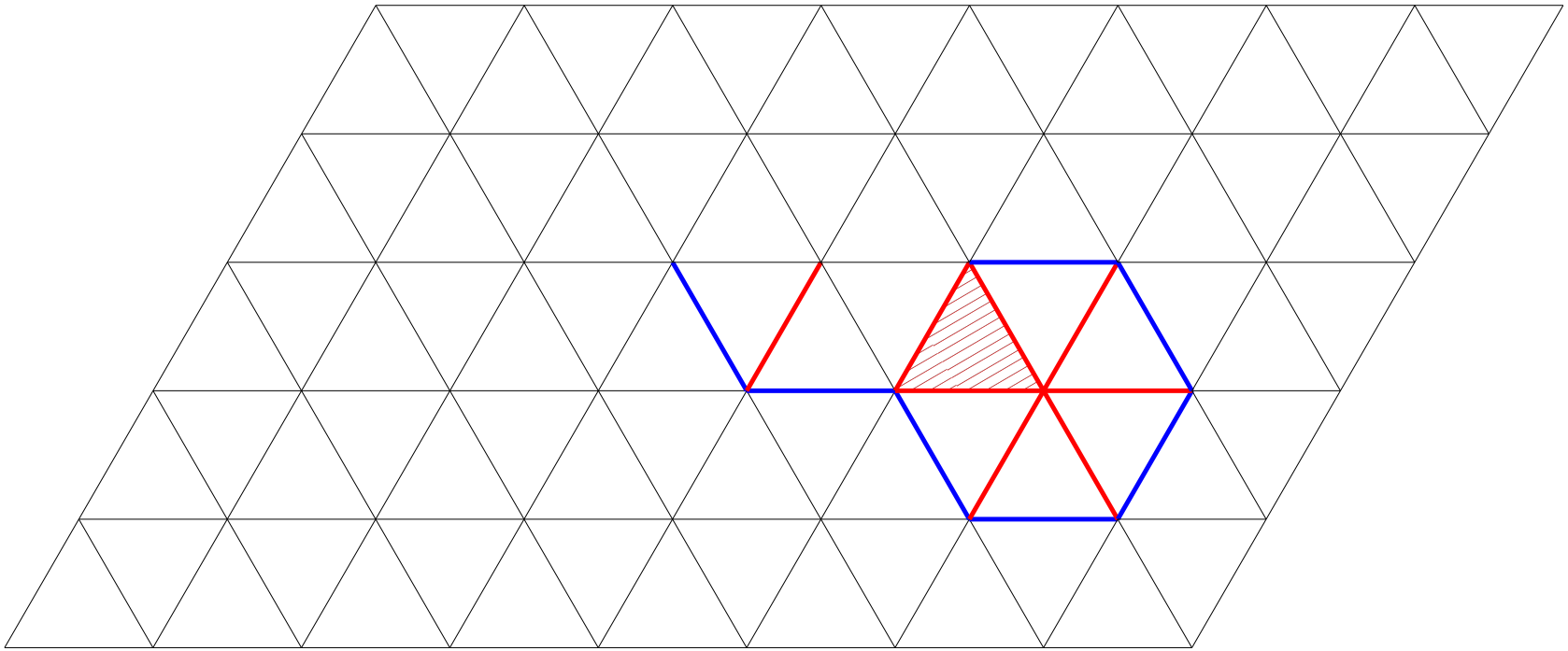
Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



Example 2: the triangle game

Alice and Bob take turns coloring unit line segments of the triangular grid, in red and blue respectively, with Alice playing first. Alice (Maker) aims at completing a unit triangle; Bob (Breaker) aims at blocking her.



- Outcome: Maker wins.

The Maker-Breaker game

- First formulated by Chvátal and Erdős (1978).
- Played on a *hypergraph* H: *vertex set* $V(H)$, *edge set* $E(H) \subseteq \mathcal{P}(V(H))$.

The Maker-Breaker game

- First formulated by Chvátal and Erdős (1978).
- Played on a *hypergraph* H: *vertex set* $V(H)$, *edge set* $E(H) \subseteq \mathcal{P}(V(H))$.
- 2-player game: Maker vs Breaker, playing alternately.
- In each round:
  - Maker picks a (not yet colored) vertex and colors it in red.
  - Breaker picks a (not yet colored) vertex and colors it in blue.
- The players' objectives:
  - Maker: get a monochromatic red edge.
  - Breaker: prevent Maker from doing so.

  Remark: no draw is possible.

The Maker-Breaker game

- First formulated by Chvátal and Erdős (1978).
- Played on a *hypergraph* H: *vertex set* $V(H)$, *edge set* $E(H) \subseteq \mathcal{P}(V(H))$.
- 2-player game: Maker vs Breaker, playing alternately.
- In each round:
  - Maker picks a (not yet colored) vertex and colors it in red.
  - Breaker picks a (not yet colored) vertex and colors it in blue.
- The players' objectives:
  - Maker: get a monochromatic red edge.
  - Breaker: prevent Maker from doing so.

  Remark: no draw is possible.
- **Maker always plays first.**

  $\rightarrow$ Only two possible outcomes: *Maker win* or *Breaker win*.

The Maker-Breaker game

- First formulated by Chvátal and Erdős (1978).
- Played on a *hypergraph* H: *vertex set* $V(H)$, *edge set* $E(H) \subseteq \mathcal{P}(V(H))$.
- 2-player game: Maker vs Breaker, playing alternately.
- In each round:
  - Maker picks a (not yet colored) vertex and colors it in red.
  - Breaker picks a (not yet colored) vertex and colors it in blue.
- The players' objectives:
  - Maker: get a monochromatic red edge.
  - Breaker: prevent Maker from doing so.

  Remark: no draw is possible.
- **Maker always plays first.**
  $\rightarrow$ Only two possible outcomes: *Maker win* or *Breaker win*.

- Question: what is the outcome on a given hypergraph, i.e. which player has a winning strategy?

The Maker-Breaker game

- First formulated by Chvátal and Erdős (1978).
- Played on a *hypergraph* H: *vertex set* $V(H)$, *edge set* $E(H) \subseteq \mathcal{P}(V(H))$.
- 2-player game: Maker vs Breaker, playing alternately.
- In each round:
  - Maker picks a (not yet colored) vertex and colors it in red.
  - Breaker picks a (not yet colored) vertex and colors it in blue.
- The players' objectives:
  - Maker: get a monochromatic red edge.
  - Breaker: prevent Maker from doing so.

  Remark: no draw is possible.
- **Maker always plays first.**
  $\rightarrow$ Only two possible outcomes: *Maker win* or *Breaker win*.

- Question: what is the outcome on a given hypergraph, i.e. which player has a winning strategy?
  $\rightarrow$ Criteria for the outcome
  $\rightarrow$ Algorithmic complexity of deciding the outcome

What if the winning sets are small?

More hope for **structural criteria** and **polynomial-time algorithms**.

What if the winning sets are small?

More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- k-*uniform* hypergraph: **each** edge is of size k.

What if the winning sets are small?

More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- k-*uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.

What if the winning sets are small?

More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- k-*uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.
- If H is 2-uniform i.e. is a graph: Maker wins if and only if H contains a P_3.

What if the winning sets are small?

More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- k-*uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.
- If H is 2-uniform i.e. is a graph: Maker wins if and only if H contains a P_3.

P_3 : 

What if the winning sets are small?

More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- *k-uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.
- If H is 2-uniform i.e. is a graph: Maker wins if and only if H contains a P_3.

P_3 : 

What if the winning sets are small?

More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- *k-uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.
- If H is 2-uniform i.e. is a graph: Maker wins if and only if H contains a P_3.

P_3 : 

What if the winning sets are small?

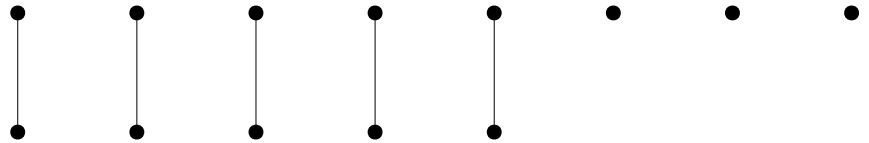
More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- *k-uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.
- If H is 2-uniform i.e. is a graph: Maker wins if and only if H contains a P_3.

P_3 :

No P_3:



What if the winning sets are small?

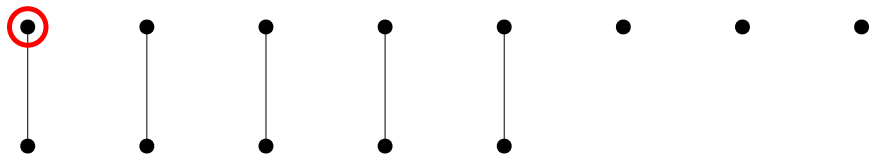
More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- *k-uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.
- If H is 2-uniform i.e. is a graph: Maker wins if and only if H contains a P_3.

P_3 :

No P_3:



What if the winning sets are small?

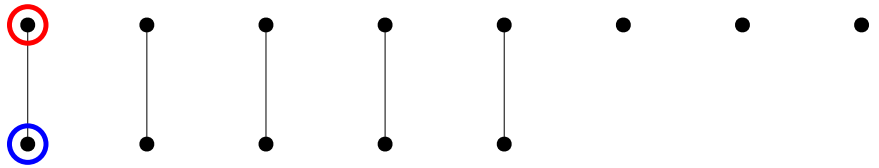
More hope for **structural criteria** and **polynomial-time algorithms**.

- Hypergraph of *rank* k: the biggest edge is of size k.
- *k-uniform* hypergraph: **each** edge is of size k.

- If H has an edge if size 1: Maker wins trivially.
- If H is 2-uniform i.e. is a graph: Maker wins if and only if H contains a P_3.

P_3 :

No P_3:

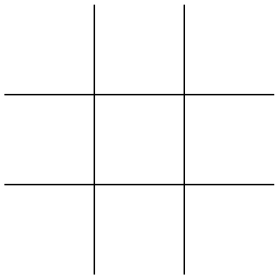


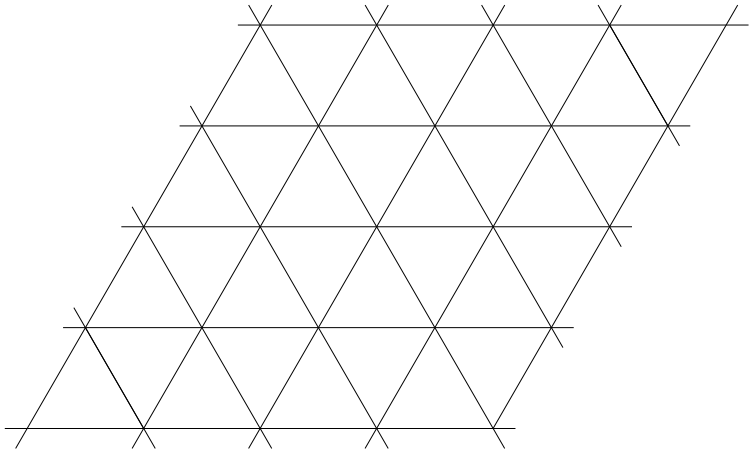
What if the winning sets are small?

- **We are interested in hypergraphs of rank 3.**

What if the winning sets are small?

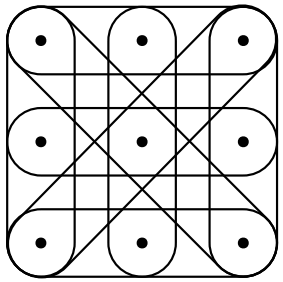
- **We are interested in hypergraphs of rank 3.**

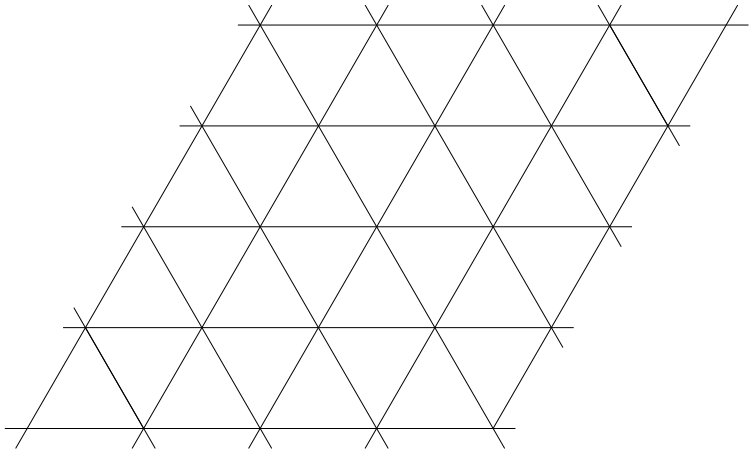


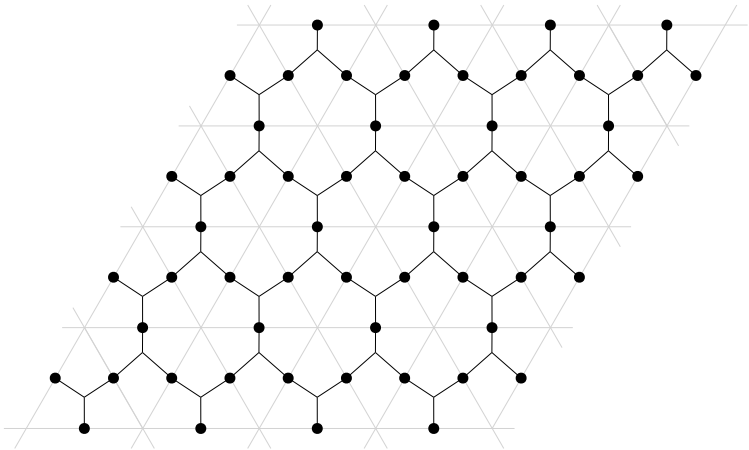


What if the winning sets are small?

- **We are interested in hypergraphs of rank 3.**

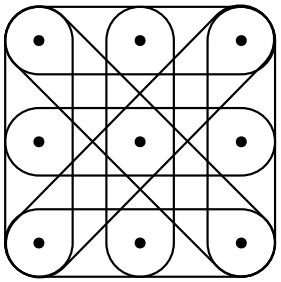


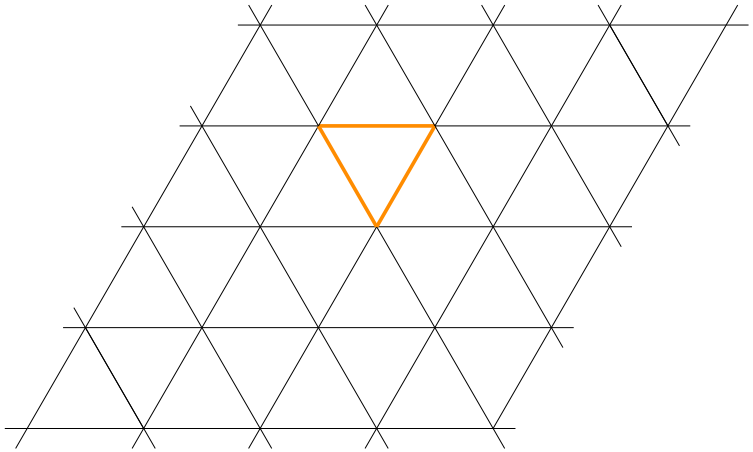


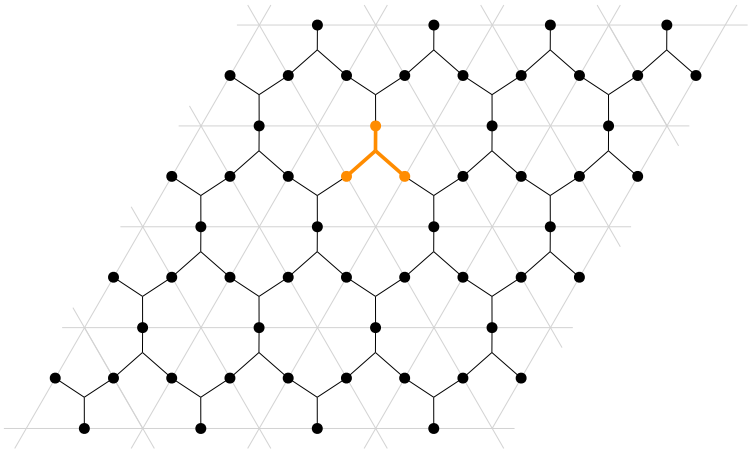


What if the winning sets are small?

- **We are interested in hypergraphs of rank 3.**

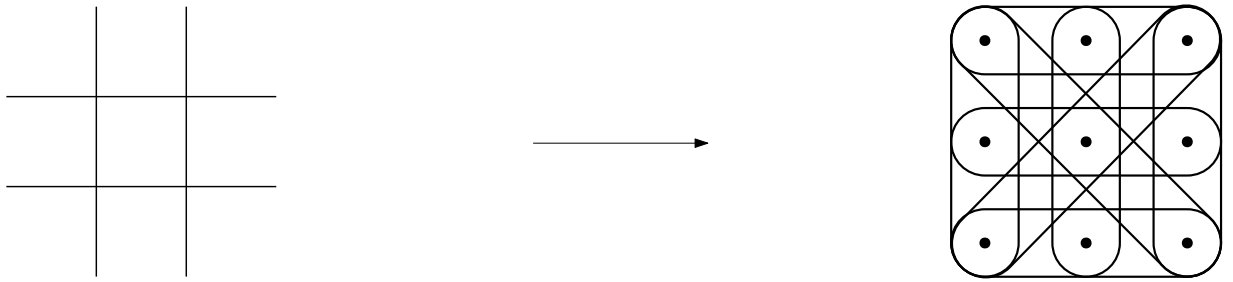


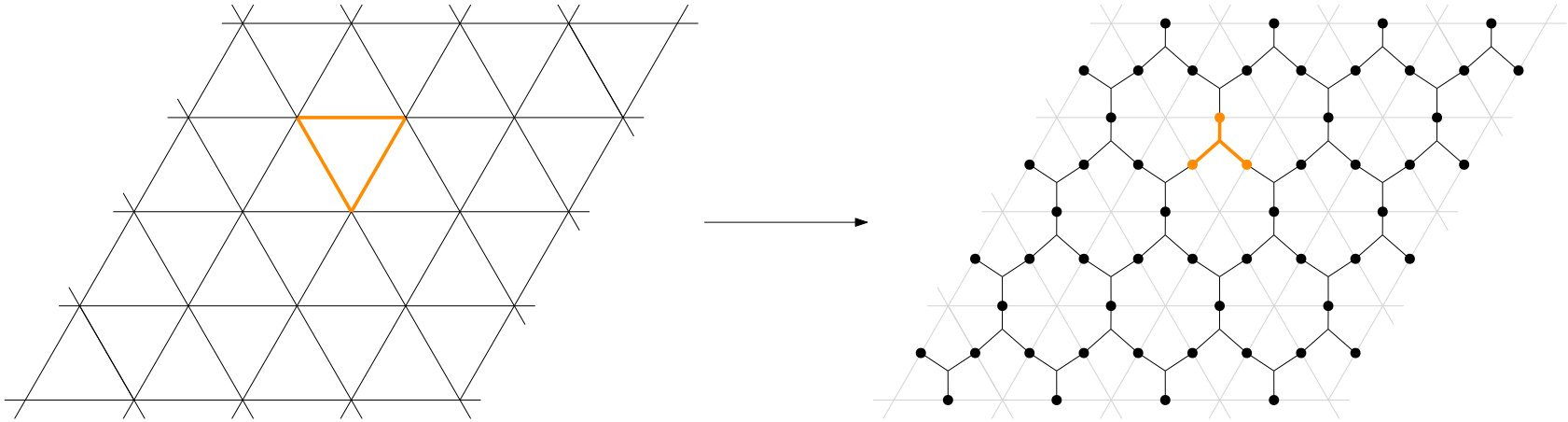




What if the winning sets are small?

- **We are interested in hypergraphs of rank 3.**





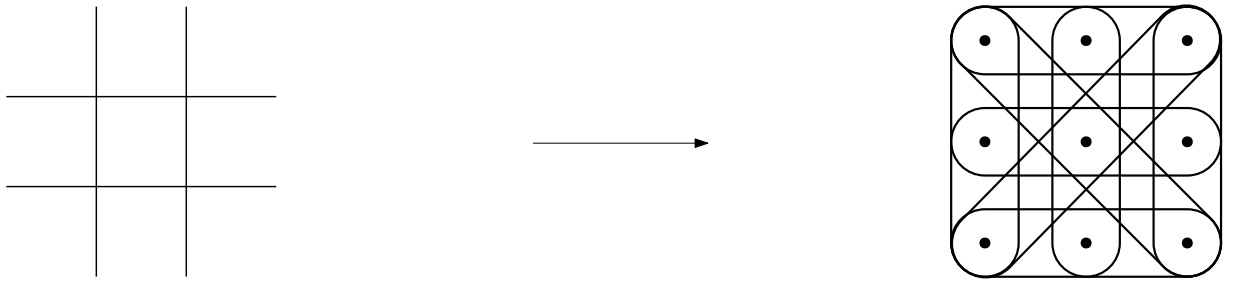
Kutz (2004) studied the *linear* subcase: $|e \cap e'| \leq 1$ for all $e \neq e'$.

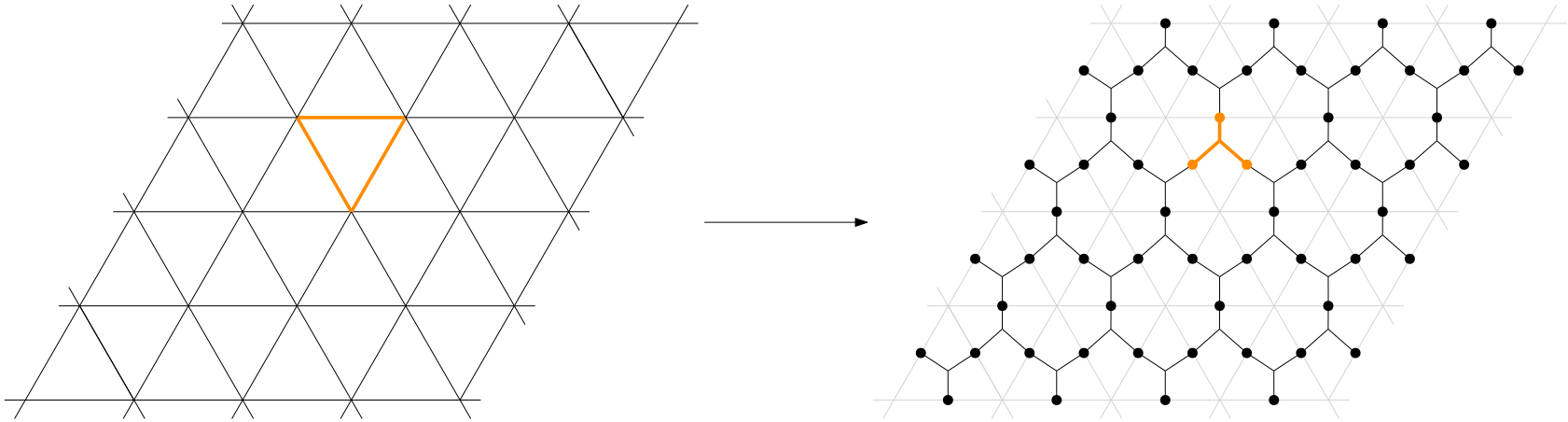
$\rightarrow$ Structural characterization for the outcome

$\rightarrow$ Polynomial-time algorithm

What if the winning sets are small?

- **We are interested in hypergraphs of rank 3.**





Kutz (2004) studied the *linear* subcase: $|e \cap e'| \leq 1$ for all $e \neq e'$.

$\rightarrow$ Structural characterization for the outcome

$\rightarrow$ Polynomial-time algorithm

We would like similar results for general hypergraphs of rank 3.

Marked hypergraphs

- In practice:
  - Maker plays a vertex x: we *mark* x.
  - Breaker plays a vertex y: we delete y i.e. we remove y as well as all edges containing y.

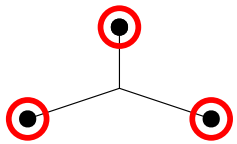
Marked hypergraphs

- In practice:
  - Maker plays a vertex x: we *mark* x.
  - Breaker plays a vertex y: we delete y i.e. we remove y as well as all edges containing y.

- We are thus playing on **marked hypergraphs**.

  Marked hypergraph: a hypergraph H with a set $M(H) \subseteq V(H)$ of *marked vertices* representing the vertices owned by Maker.

  Maker wins if and only if some edge has all its vertices marked.



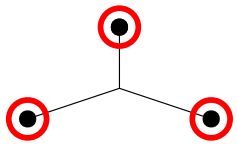
Marked hypergraphs

- In practice:
  - Maker plays a vertex x: we *mark* x.
  - Breaker plays a vertex y: we delete y i.e. we remove y as well as all edges containing y.

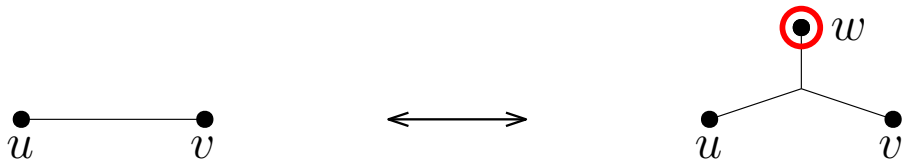
- We are thus playing on **marked hypergraphs**.

  Marked hypergraph: a hypergraph H with a set $M(H) \subseteq V(H)$ of *marked vertices* representing the vertices owned by Maker.

  Maker wins if and only if some edge has all its vertices marked.

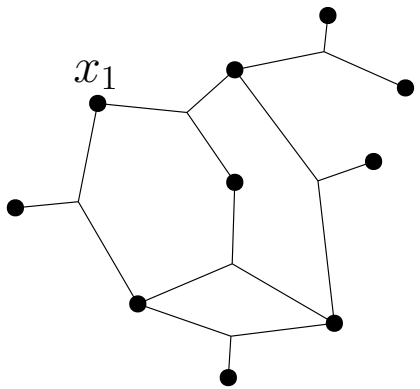


- Hypergraph of rank 3 $\longleftrightarrow$ 3-uniform marked hypergraph.

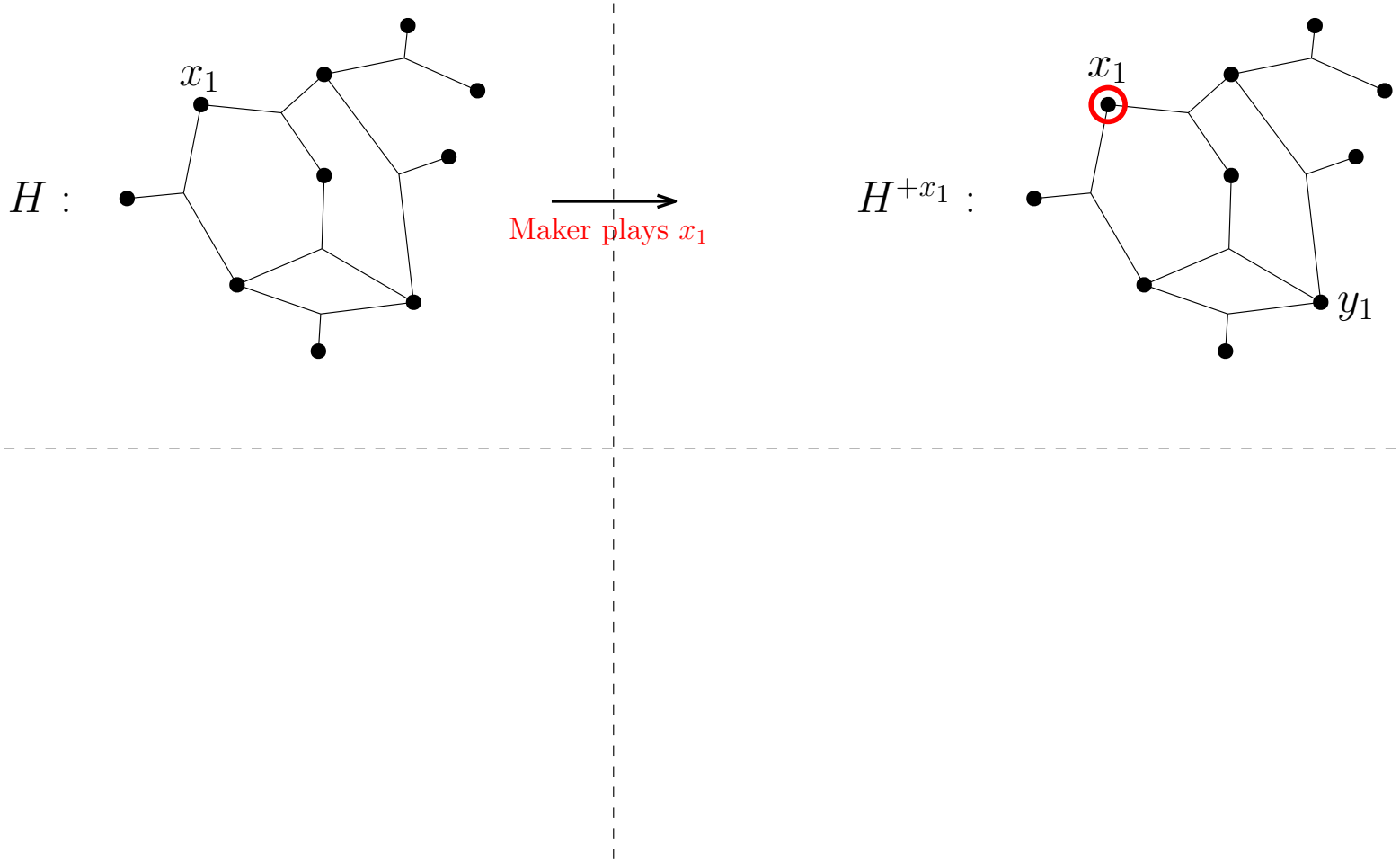


Progression of a game: example

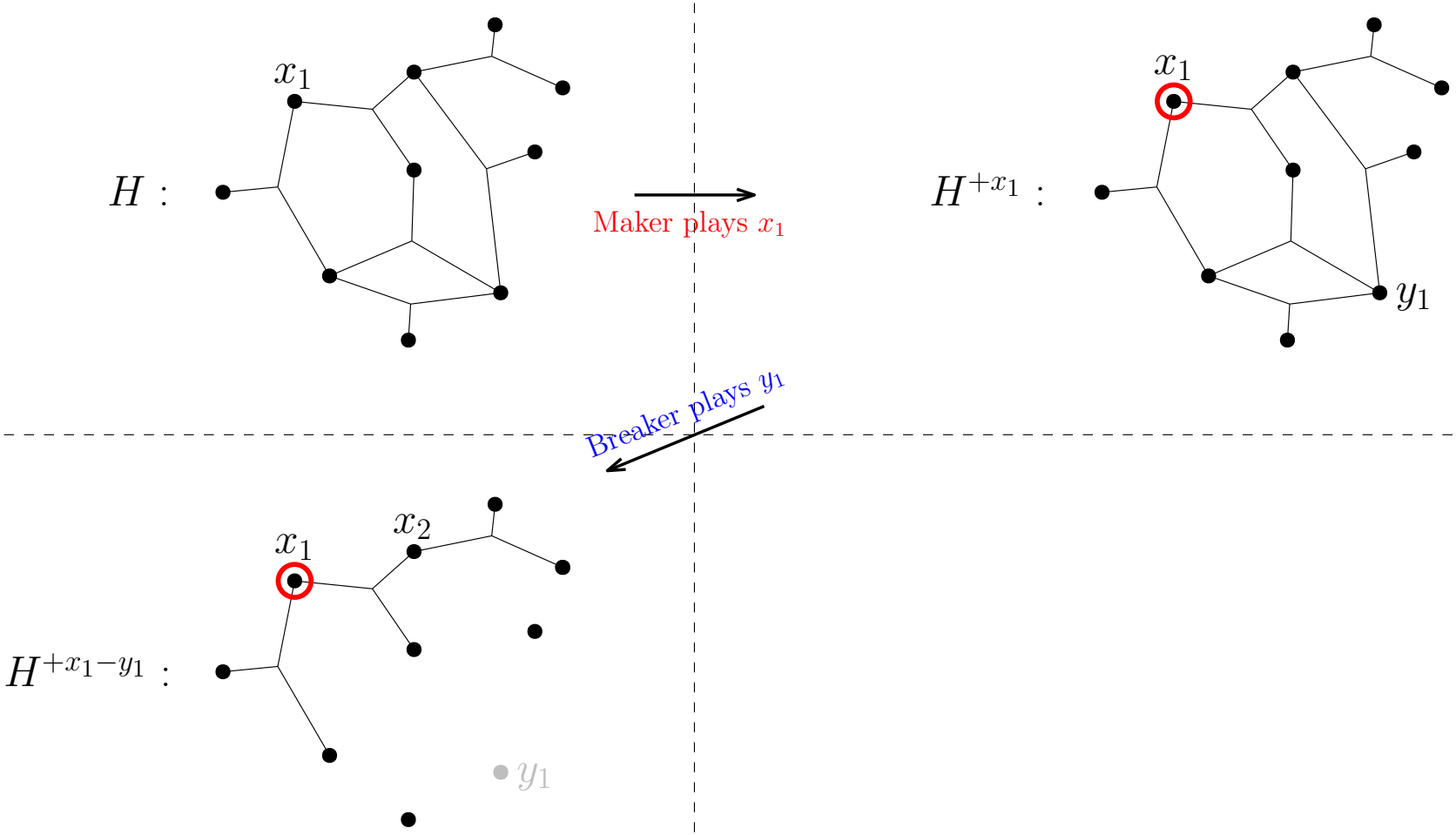
H :



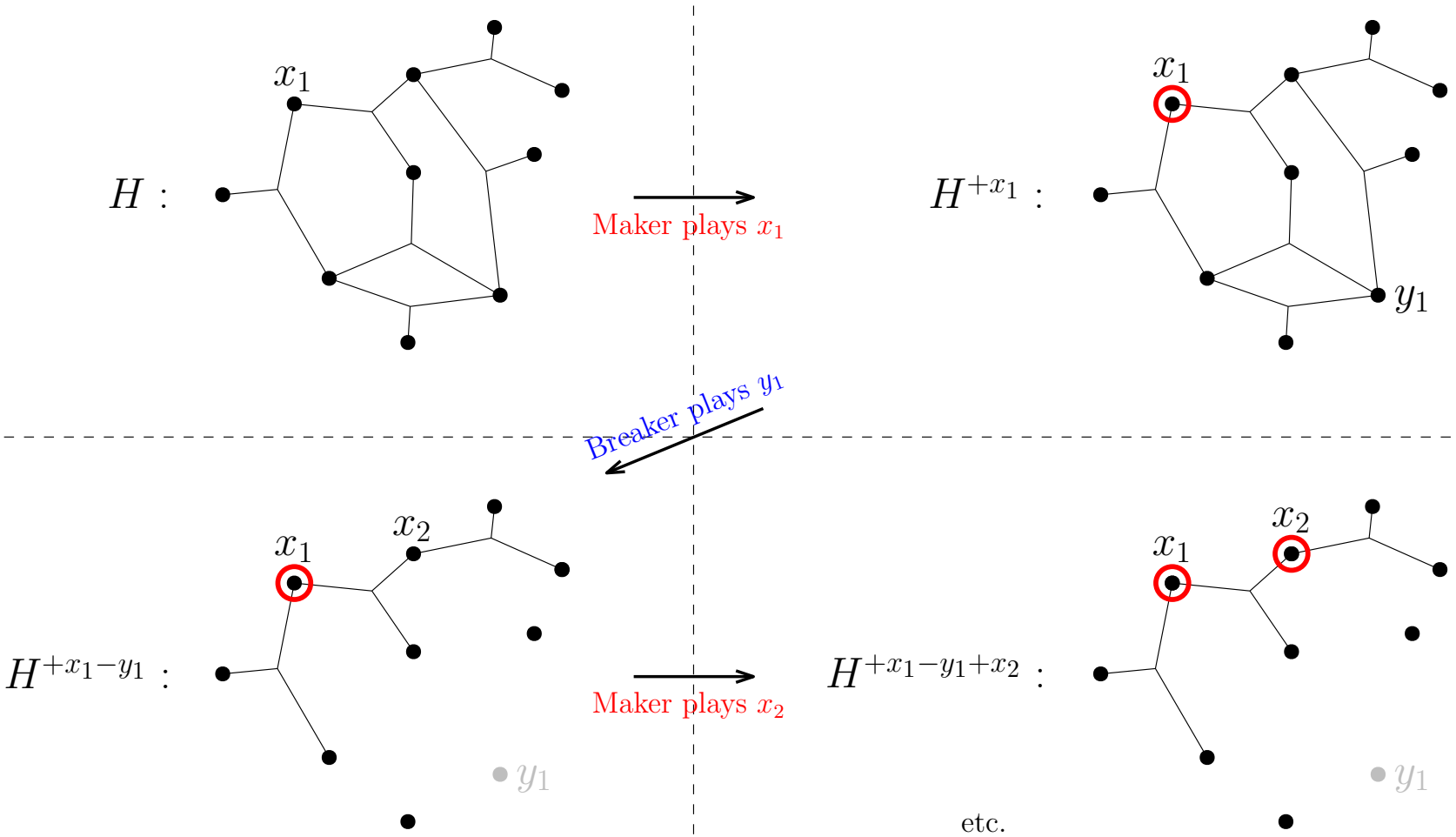
Progression of a game: example



Progression of a game: example

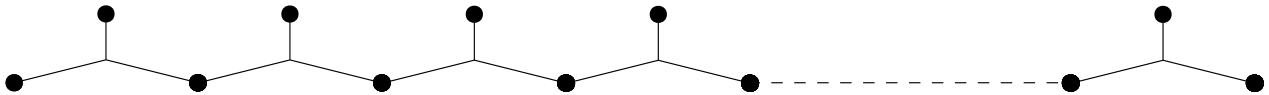


Progression of a game: example



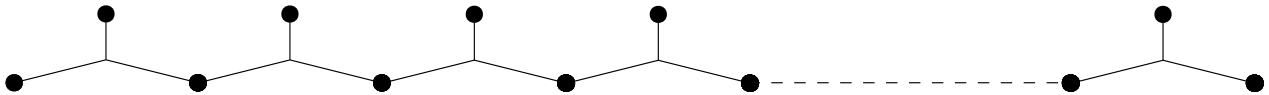
The forcing principle

Chain: linear simple path.

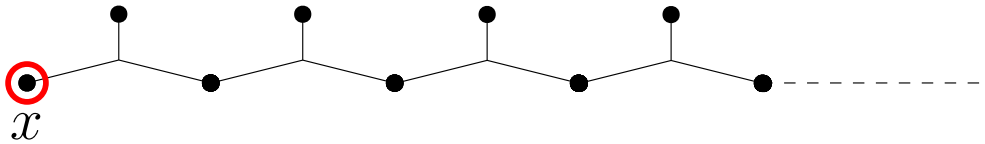


The forcing principle

Chain: linear simple path.

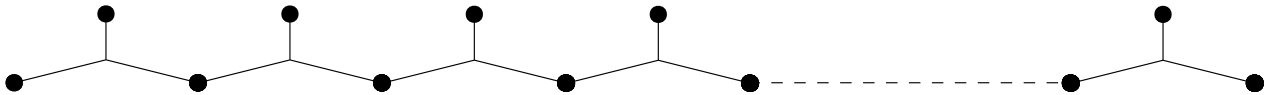


Along a chain with a marked extremity, Maker can engage a forcing sequence.

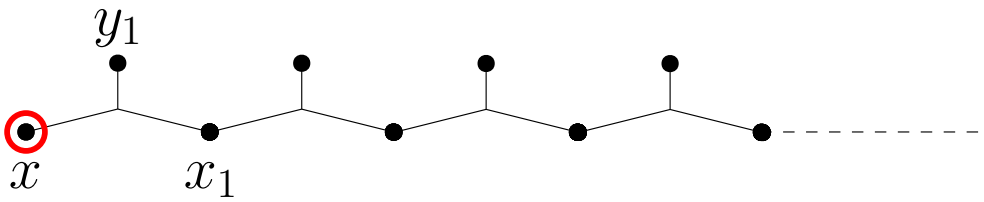


The forcing principle

Chain: linear simple path.

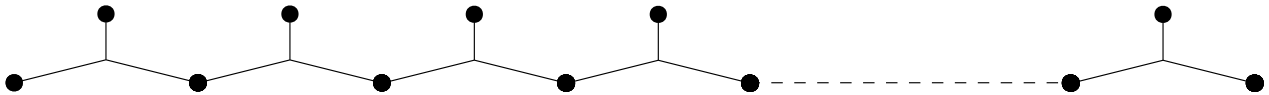


Along a chain with a marked extremity, Maker can engage a forcing sequence.

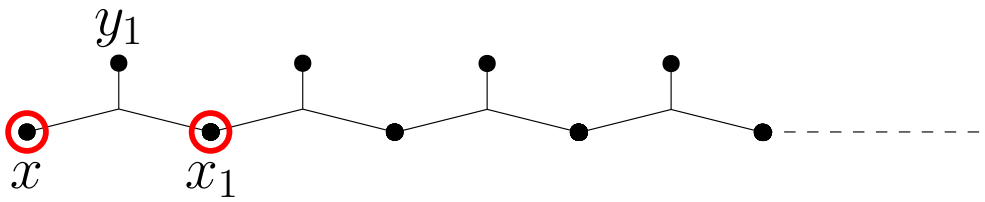


The forcing principle

Chain: linear simple path.

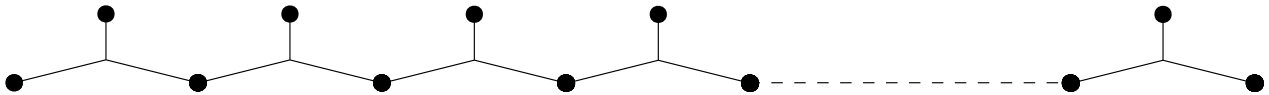


Along a chain with a marked extremity, Maker can engage a forcing sequence.

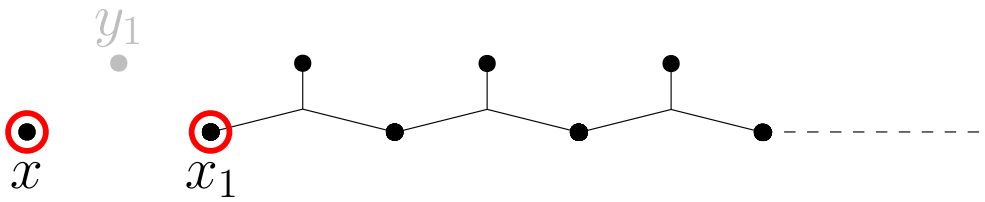


The forcing principle

Chain: linear simple path.

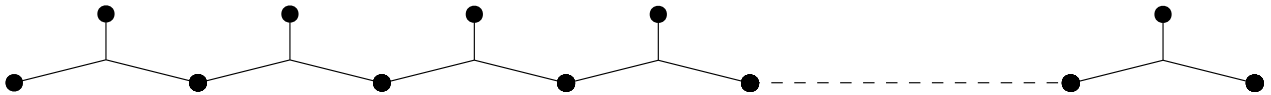


Along a chain with a marked extremity, Maker can engage a forcing sequence.

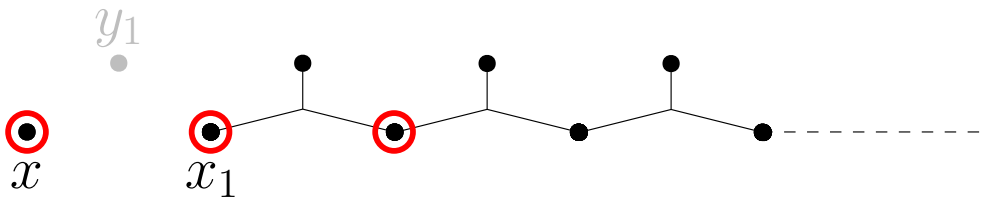


The forcing principle

Chain: linear simple path.

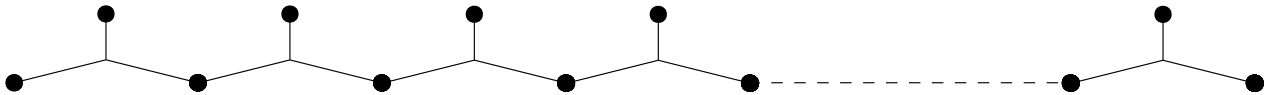


Along a chain with a marked extremity, Maker can engage a forcing sequence.

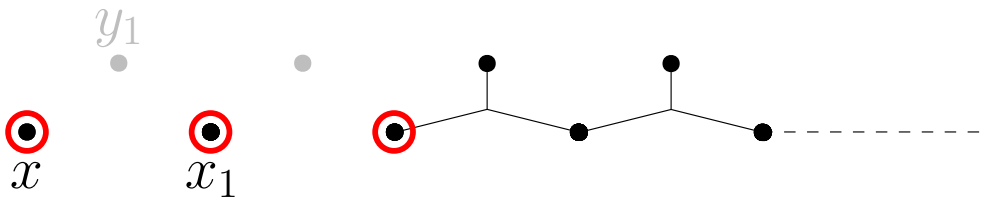


The forcing principle

Chain: linear simple path.

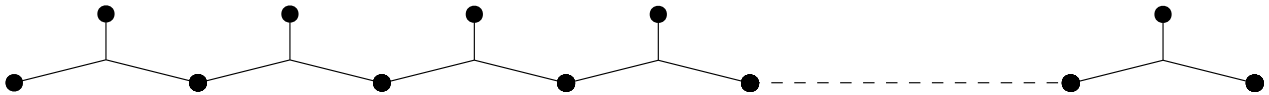


Along a chain with a marked extremity, Maker can engage a forcing sequence.

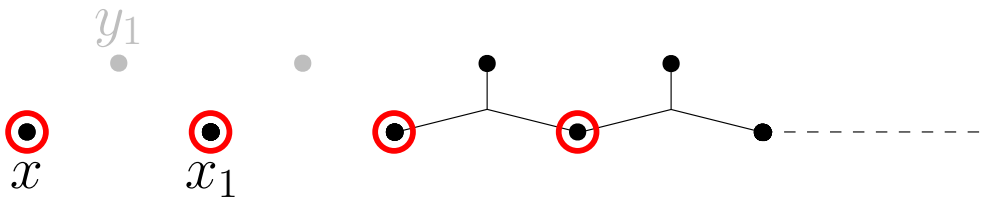


The forcing principle

Chain: linear simple path.

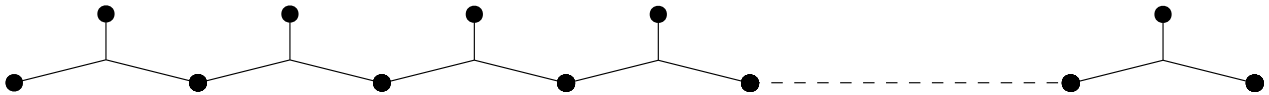


Along a chain with a marked extremity, Maker can engage a forcing sequence.

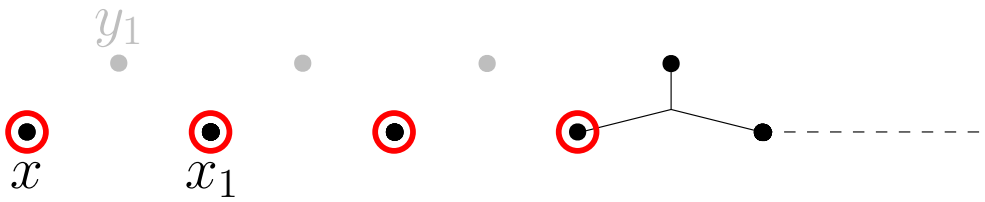


The forcing principle

Chain: linear simple path.

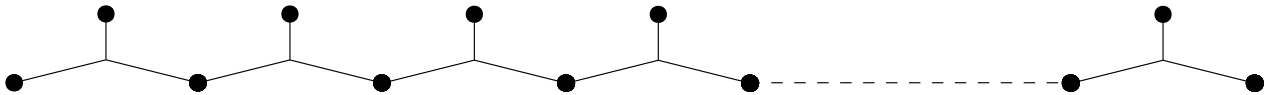


Along a chain with a marked extremity, Maker can engage a forcing sequence.

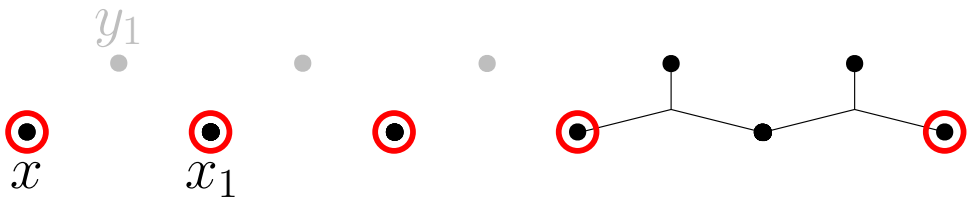


The forcing principle

Chain: linear simple path.

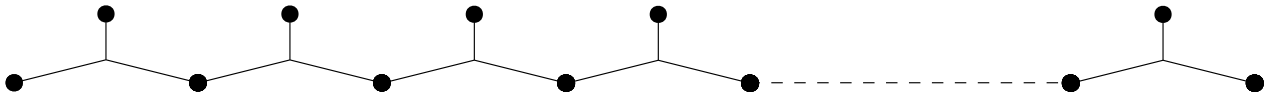


Along a chain with a marked extremity, Maker can engage a forcing sequence.

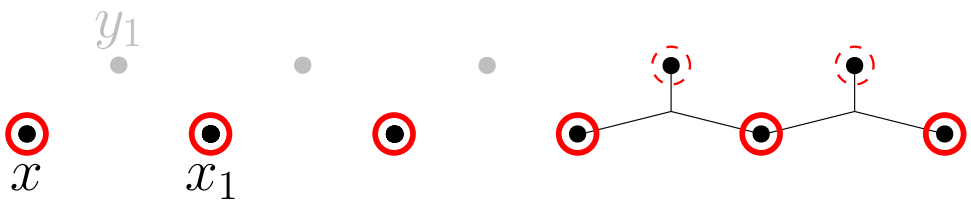


The forcing principle

Chain: linear simple path.

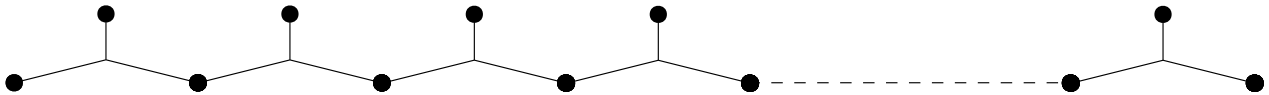


Along a chain with a marked extremity, Maker can engage a forcing sequence.



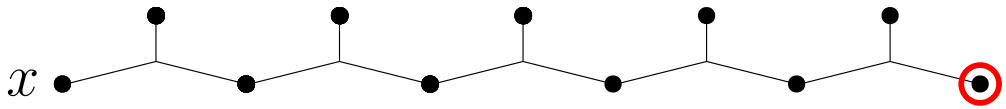
The forcing principle

Chain: linear simple path.



Along a chain with a marked extremity, Maker can engage a forcing sequence.

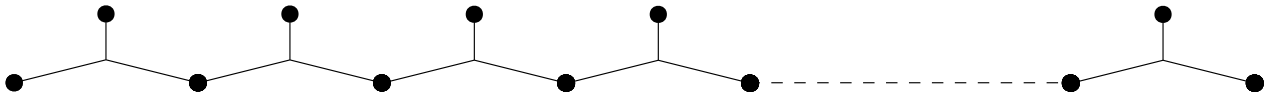
- *x-snake*:



  If Maker plays x, then Breaker is forced to answer inside the x-snake.

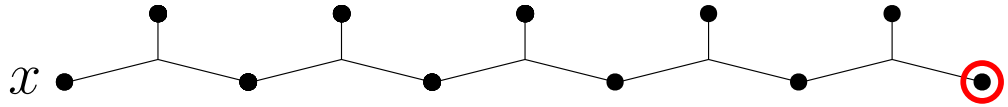
The forcing principle

Chain: linear simple path.



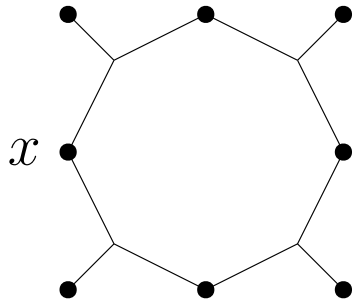
Along a chain with a marked extremity, Maker can engage a forcing sequence.

- *x-snake*:



If Maker plays x, then Breaker is forced to answer inside the x-snake.

- *x-cycle*:



If Maker plays x, then Breaker is forced to answer inside the x-cycle.

Dangers

We adopt **Breaker**'s point of view.

Dangers

We adopt **Breaker**'s point of view.

Danger at x in H: a subhypergraph D of H containing x such that D^{+x} is a Maker win.
We then say the pair (D, x) is a *danger*. If $\mathcal{F}$ is a family of dangers, then an element of $\mathcal{F}$ is called an *$\mathcal{F}$-danger*.

Dangers

We adopt **Breaker**'s point of view.

Danger at x in H: a subhypergraph D of H containing x such that D^{+x} is a Maker win.
We then say the pair (D, x) is a *danger*. If $\mathcal{F}$ is a family of dangers, then an element of $\mathcal{F}$ is called an *$\mathcal{F}$-danger*.

If Maker plays x then Breaker is **forced** to answer in $V(D)$ immediately.

Dangers

We adopt **Breaker**'s point of view.

Danger at x in H: a subhypergraph D of H containing x such that D^{+x} is a Maker win.
We then say the pair (D, x) is a *danger*. If $\mathcal{F}$ is a family of dangers, then an element of $\mathcal{F}$ is called an *$\mathcal{F}$-danger*.

If Maker plays x then Breaker is **forced** to answer in $V(D)$ immediately.

Danger intersection property

Let $\mathcal{F}$ be a family of dangers. We say a marked hypergraph H has property $J(\mathcal{F})$ when, for all non-marked vertex x of H, the intersection of the $\mathcal{F}$-dangers at x in H is **non-empty**.

(where the intersection excludes x itself and all marked vertices)

Dangers

We adopt **Breaker**'s point of view.

Danger at x in H: a subhypergraph D of H containing x such that D^{+x} is a Maker win.
We then say the pair (D, x) is a *danger*. If $\mathcal{F}$ is a family of dangers, then an element of $\mathcal{F}$ is called an *$\mathcal{F}$-danger*.

If Maker plays x then Breaker is **forced** to answer in $V(D)$ immediately.

Danger intersection property

Let $\mathcal{F}$ be a family of dangers. We say a marked hypergraph H has property $J(\mathcal{F})$ when, for all non-marked vertex x of H, the intersection of the $\mathcal{F}$-dangers at x in H is **non-empty**.

(where the intersection excludes x itself and all marked vertices)

Necessary condition for a Breaker win

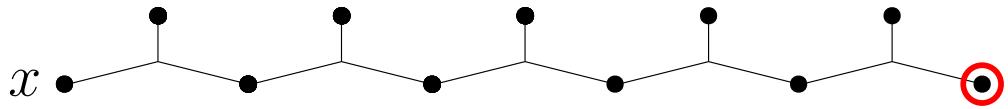
Let $\mathcal{F}$ be a family of dangers, and let H be a marked hypergraph. If H is a Breaker win the H has property $J(\mathcal{F})$.

Elementary dangers in the 3-uniform case

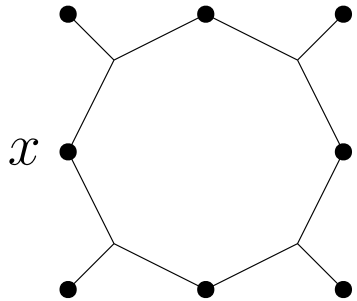
Family of elementary dangers

$\mathcal{D}_0 :=$ family of all snakes and cycles.

- *x-snake*:

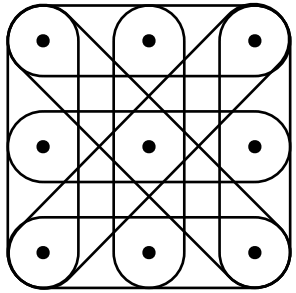


- *x-cycle*:



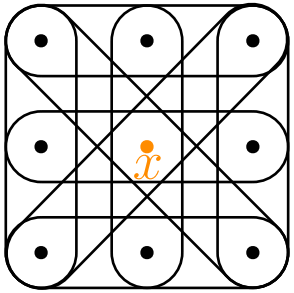
An example: back to asymmetrical tic-tac-toe

$H =$ tic-tac-toe hypergraph:



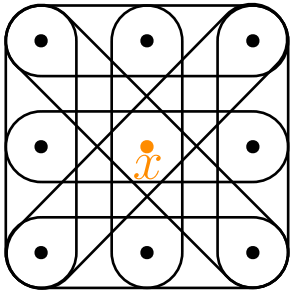
An example: back to asymmetrical tic-tac-toe

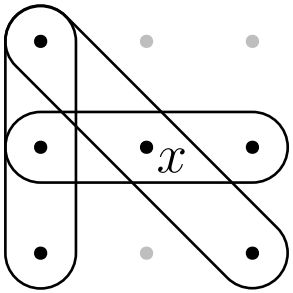
$H =$ tic-tac-toe hypergraph:

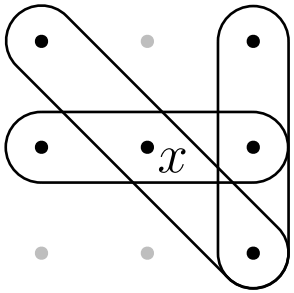


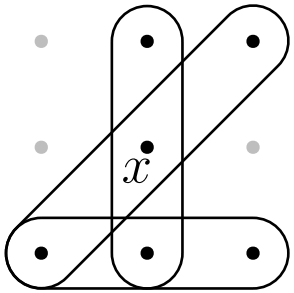
An example: back to asymmetrical tic-tac-toe

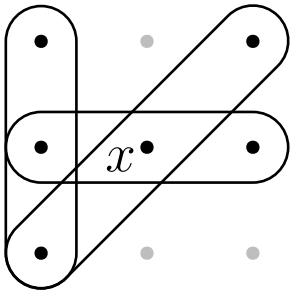
H = tic-tac-toe hypergraph:







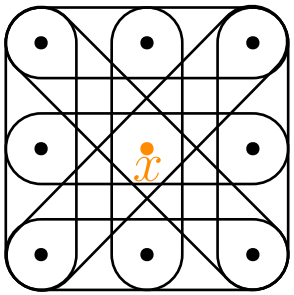


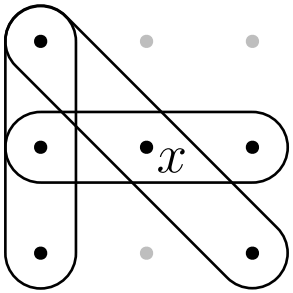


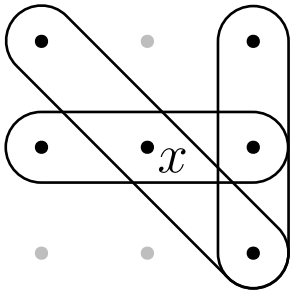
These four x-cycles are $\mathcal{D}_0$-dangers at x, and their intersection is empty.

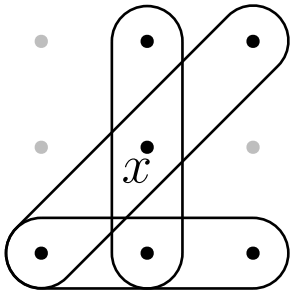
An example: back to asymmetrical tic-tac-toe

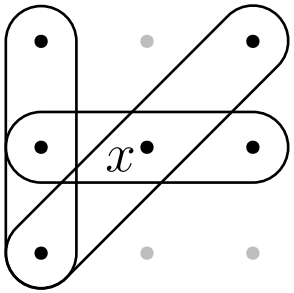
H = tic-tac-toe hypergraph:











These four x-cycles are $\mathcal{D}_0$-dangers at x, and their intersection is empty.

$\Longrightarrow$ H does not have property $J(\mathcal{D}_0)$, so H is a **Maker win**.

Maker can win with first move x.

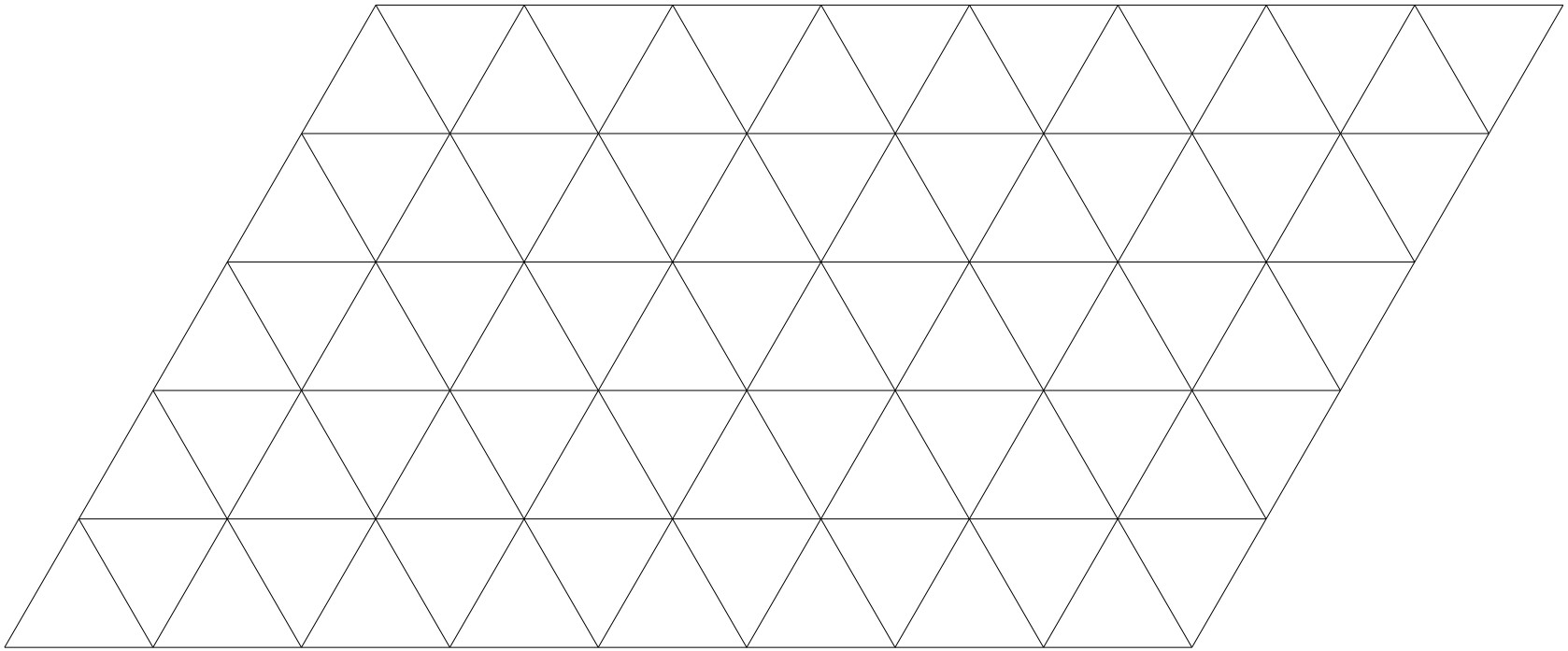
A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

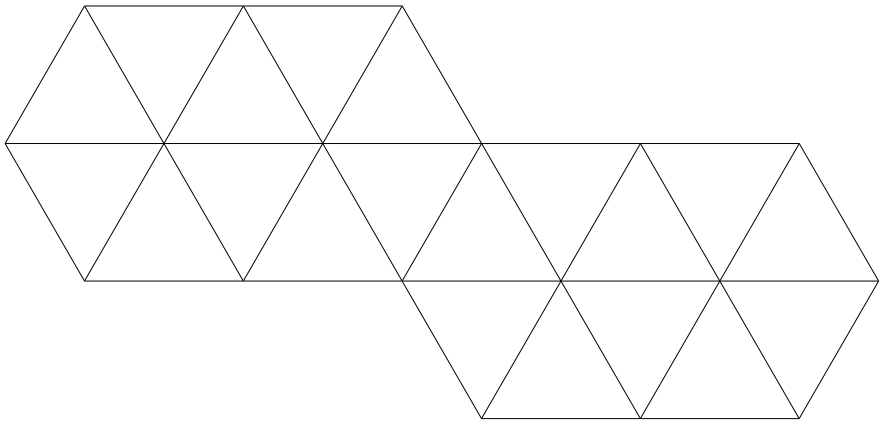
The answer is no. A counterexample:



A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

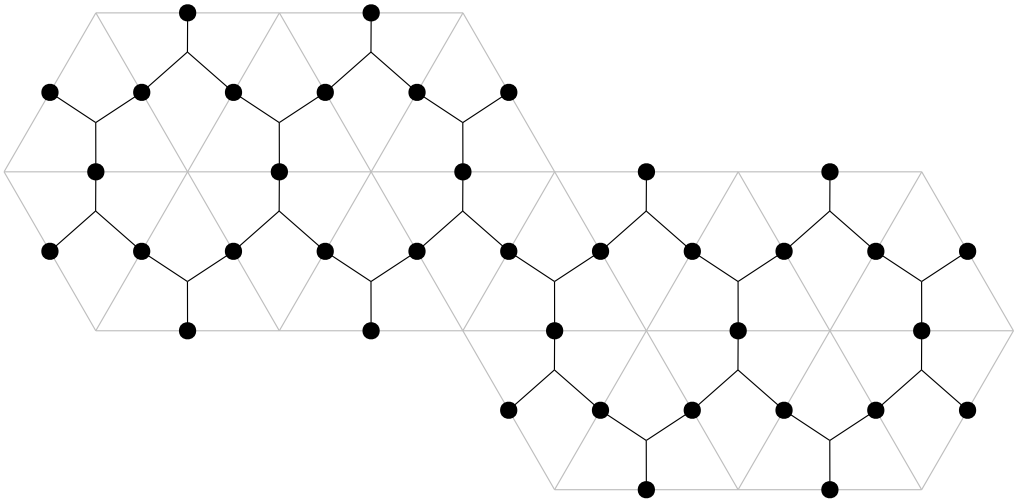


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H

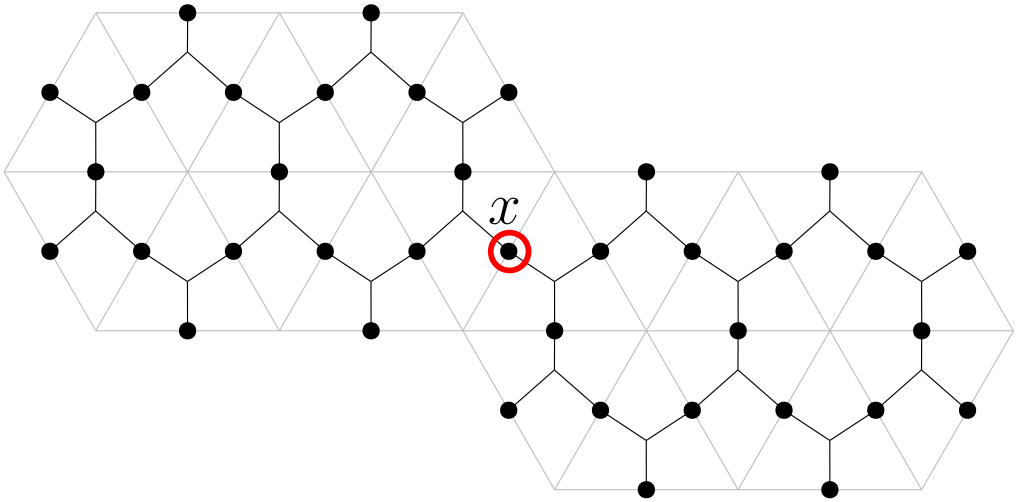


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H^{+x}

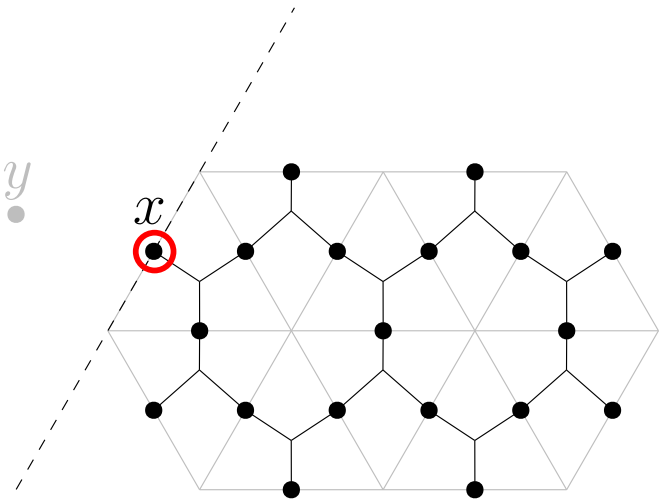


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H^{+x-y}

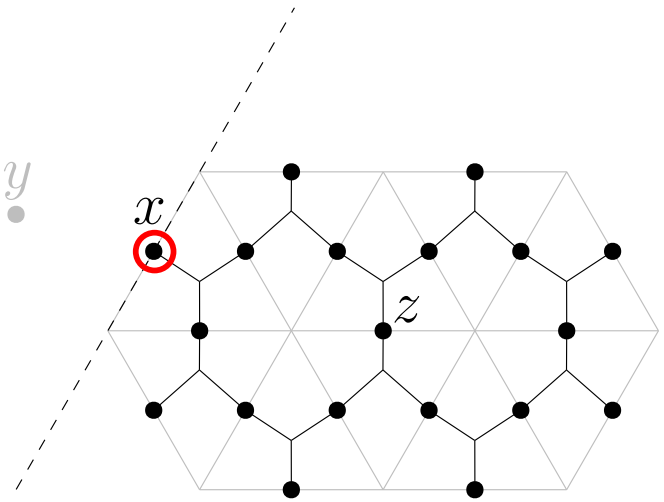


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H^{+x-y}

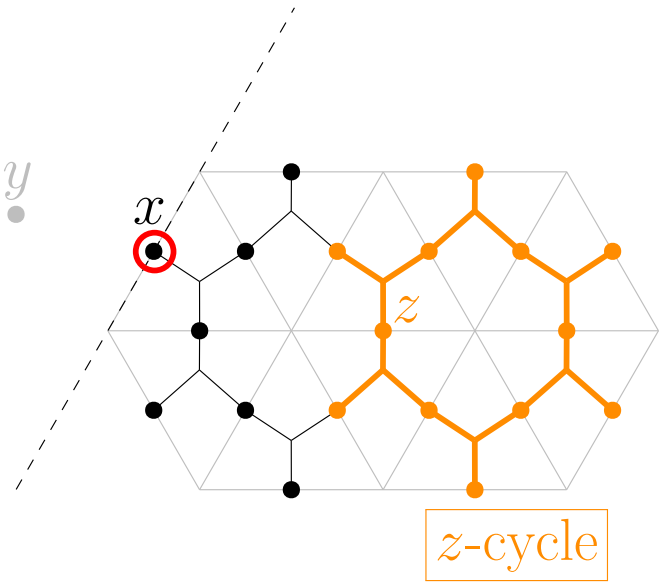


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H^{+x-y}

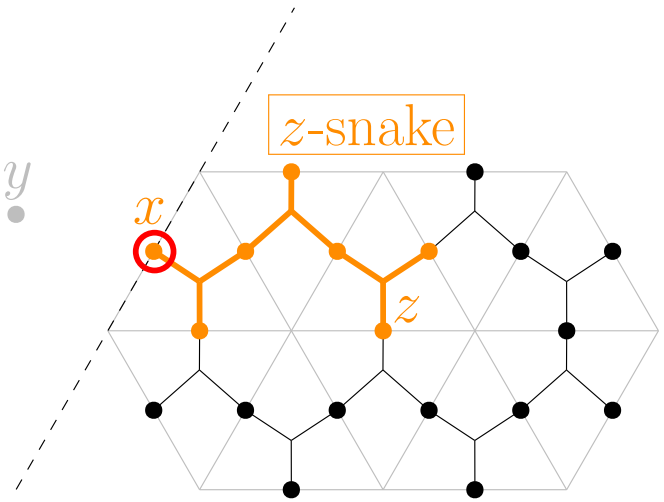


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H^{+x-y}

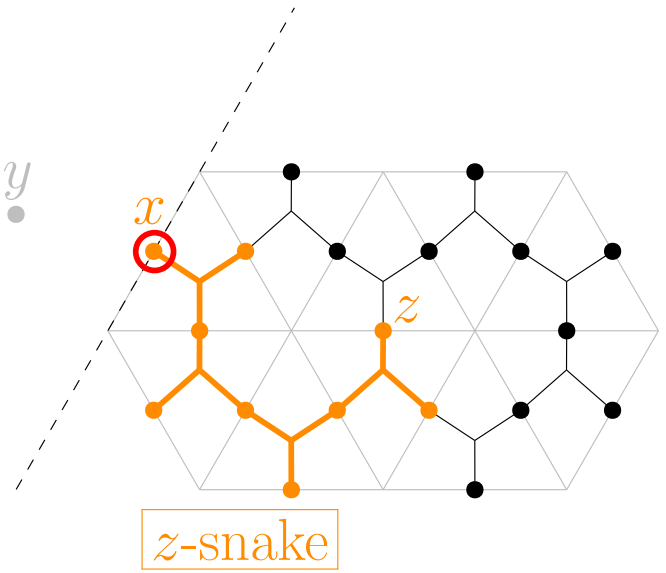


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H^{+x-y}

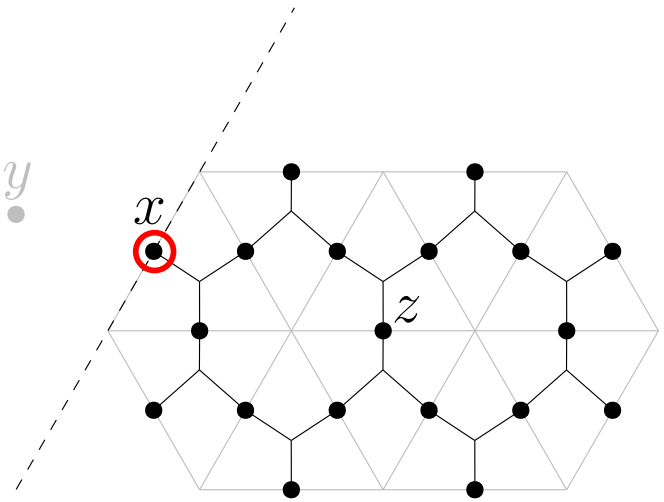


A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H^{+x-y}



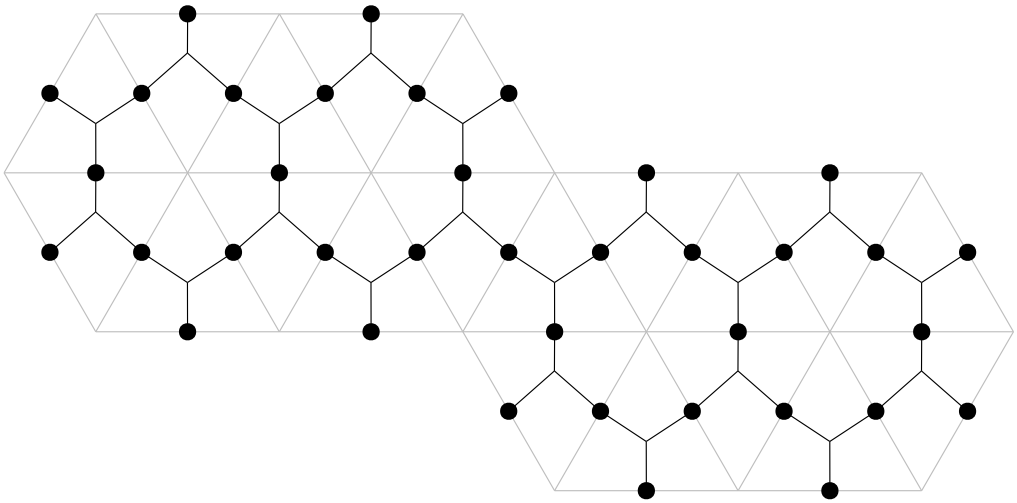
- H has property $J(\mathcal{D}_0)$.
- H^{+x-y} does not have property $J(\mathcal{D}_0)$ (violated by z).
  $\Longrightarrow$ H^{+x-y} is a Maker win, so H is a Maker win.

A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H



- H has property $J(\mathcal{D}_0)$.
- H^{+x-y} does not have property $J(\mathcal{D}_0)$ (violated by z).
  $\Longrightarrow$ H^{+x-y} is a Maker win, so H is a Maker win.

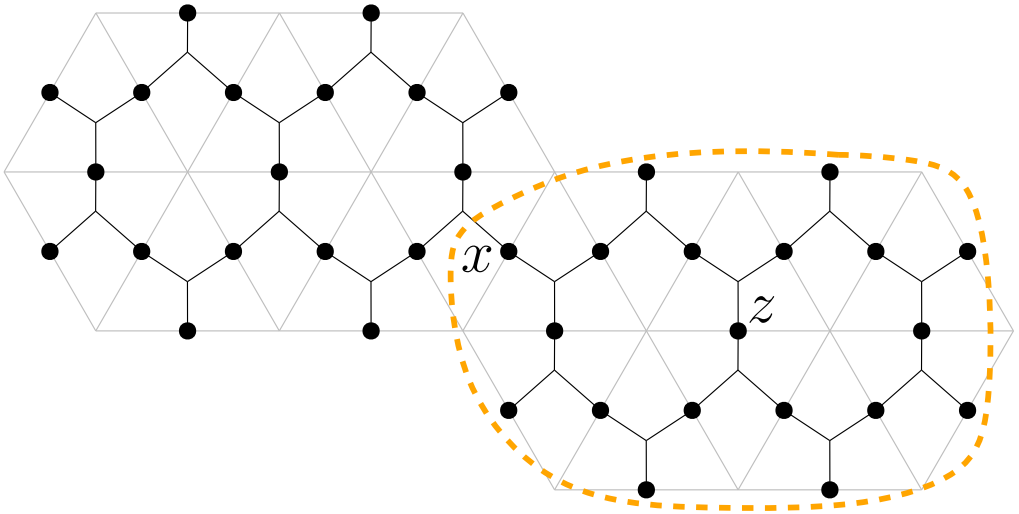
- Property $J(\mathcal{D}_0)$ ensures that Breaker can destroy the $\mathcal{D}_0$-dangers **in the first round**, but it says nothing about subsequent rounds!

A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H



- H has property $J(\mathcal{D}_0)$.
- H^{+x-y} does not have property $J(\mathcal{D}_0)$ (violated by z).
  $\Longrightarrow$ H^{+x-y} is a Maker win, so H is a Maker win.

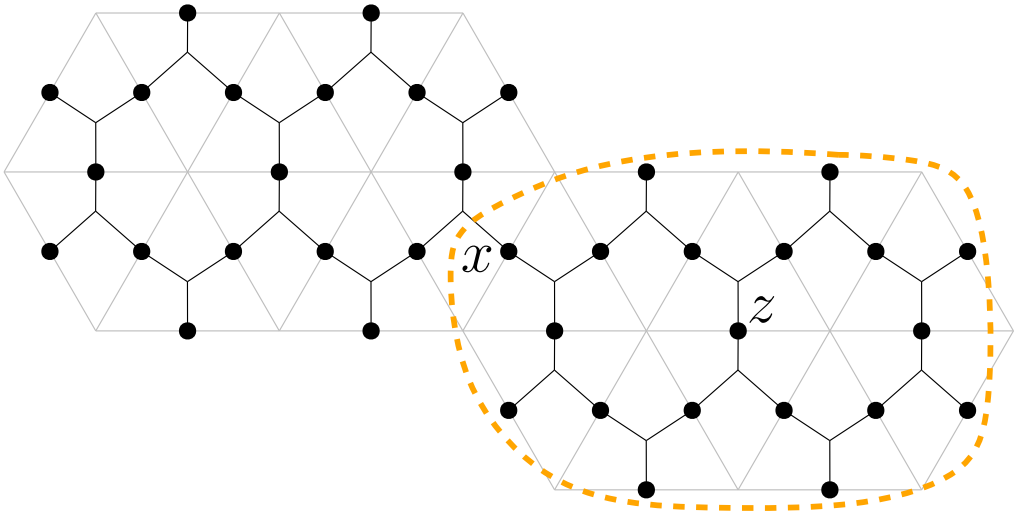
- Property $J(\mathcal{D}_0)$ ensures that Breaker can destroy the $\mathcal{D}_0$-dangers **in the first round**, but it says nothing about subsequent rounds!

A sufficient condition?

Is property $J(\mathcal{D}_0)$ necessary **and sufficient** in the 3-uniform case?

The answer is no. A counterexample:

H



- H has property $J(\mathcal{D}_0)$.
- H^{+x-y} does not have property $J(\mathcal{D}_0)$ (violated by z).
  $\Longrightarrow$ H^{+x-y} is a Maker win, so H is a Maker win.

- Property $J(\mathcal{D}_0)$ ensures that Breaker can destroy the $\mathcal{D}_0$-dangers **in the first round**, but it says nothing about subsequent rounds!
- We are going to **extend** our family of dangers.

Danger prevention: thinking ahead

Let $\mathcal{F}$ be a family of dangers.

Construction of $\mathcal{F}^*$ from $\mathcal{F}$

We add all (D, x) such that D^{+x} is a **union** of $\mathcal{F}$-dangers at some common vertex z whose intersection is empty.
(where the intersection excludes x and z themselves as well as all marked vertices)

Danger prevention: thinking ahead

Let $\mathcal{F}$ be a family of dangers.

Construction of $\mathcal{F}^*$ from $\mathcal{F}$

We add all (D, x) such that D^{+x} is a **union** of $\mathcal{F}$-dangers at some common vertex z whose intersection is empty.
(where the intersection excludes x and z themselves as well as all marked vertices)

$J(\mathcal{F}) =$ Breaker can destroy the $\mathcal{F}$-dangers **in the first round**

$J(\mathcal{F}^*) =$ Breaker can destroy the $\mathcal{F}^*$-dangers in the first round
$=$ Breaker can destroy the $\mathcal{F}$-dangers **in the first two rounds**

Danger prevention: thinking ahead

Let $\mathcal{F}$ be a family of dangers.

Construction of $\mathcal{F}^*$ from $\mathcal{F}$

We add all (D, x) such that D^{+x} is a **union** of $\mathcal{F}$-dangers at some common vertex z whose intersection is empty.
(where the intersection excludes x and z themselves as well as all marked vertices)

$J(\mathcal{F}) =$ Breaker can destroy the $\mathcal{F}$-dangers **in the first round**

$J(\mathcal{F}^*) =$ Breaker can destroy the $\mathcal{F}^*$-dangers in the first round
$=$ Breaker can destroy the $\mathcal{F}$-dangers **in the first two rounds**

$J(\mathcal{F}^{*r}) =$ Breaker can destroy the $\mathcal{F}^{*r}$-dangers in the first round
$=$ Breaker can destroy the $\mathcal{F}$-dangers **in the first $r+1$ rounds**

Danger prevention: thinking ahead

Let $\mathcal{F}$ be a family of dangers.

Construction of $\mathcal{F}^*$ from $\mathcal{F}$

We add all (D, x) such that D^{+x} is a **union** of $\mathcal{F}$-dangers at some common vertex z whose intersection is empty.
(where the intersection excludes x and z themselves as well as all marked vertices)

$J(\mathcal{F}) =$ Breaker can destroy the $\mathcal{F}$-dangers **in the first round**

$J(\mathcal{F}^*) =$ Breaker can destroy the $\mathcal{F}^*$-dangers in the first round
$=$ Breaker can destroy the $\mathcal{F}$-dangers **in the first two rounds**

$J(\mathcal{F}^{*r}) =$ Breaker can destroy the $\mathcal{F}^{*r}$-dangers in the first round
$=$ Breaker can destroy the $\mathcal{F}$-dangers **in the first $r+1$ rounds**

$$\begin{array}{ccccccc} \mathcal{F} & \subseteq & \mathcal{F}^* & \subseteq & \mathcal{F}^{**} & \subseteq & \mathcal{F}^{***} \subseteq \quad \dots \\ J(\mathcal{F}) & \Longleftarrow & J(\mathcal{F}^*) & \Longleftarrow & J(\mathcal{F}^{**}) & \Longleftarrow & J(\mathcal{F}^{***}) \Longleftarrow \quad \dots \end{array}$$

Main structural result

The property $J(\mathcal{D}_0^{*r})$ is necessary for Breaker to win.

For which r does it become **sufficient** in the 3-uniform case?

Main structural result

The property $J(\mathcal{D}_0^{*r})$ is necessary for Breaker to win.

For which r does it become **sufficient** in the 3-uniform case?
$\rightarrow$ The answer is $r = 2$.

Main structural result

The property $J(\mathcal{D}_0^{*r})$ is necessary for Breaker to win.

For which r does it become **sufficient** in the 3-uniform case?
$\rightarrow$ The answer is $r = 2$.

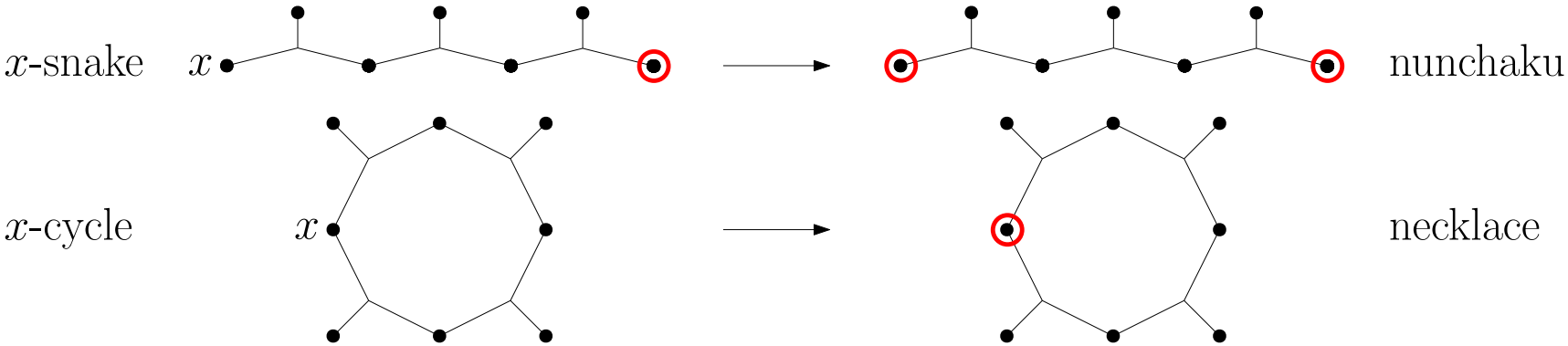
Theorem [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Breaker win if and only if H has property $J(\mathcal{D}_0^{**})$. More specifically:

- If H does not have property $J(\mathcal{D}_0^{**})$ then any x such that the $\mathcal{D}_0^{**}$-dangers at x do not intersect is a winning first move for Maker.
- If H has property $J(\mathcal{D}_0^{**})$ then, for all first move x of Maker, any y in the intersection of the $\mathcal{D}_0^{**}$-dangers at x is a winning answer for Breaker.

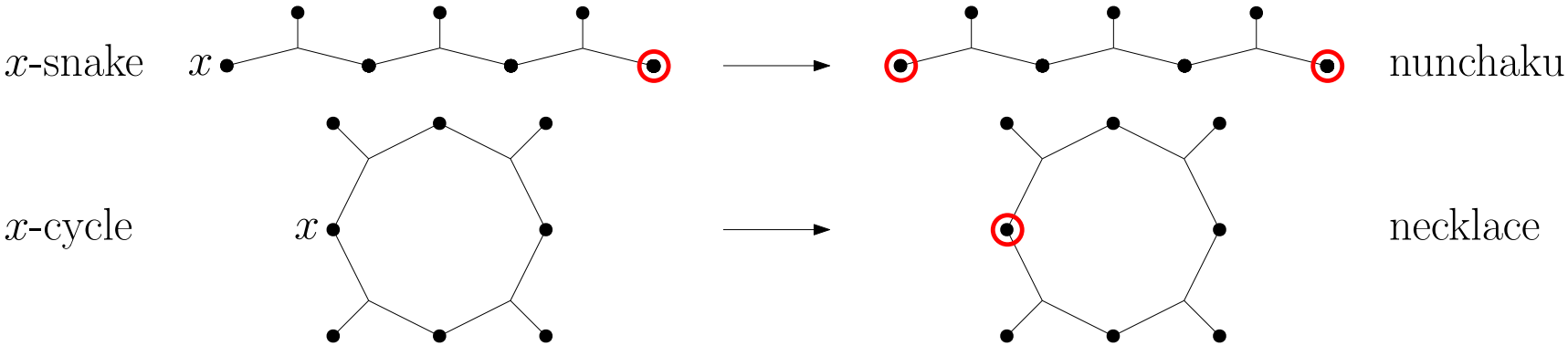
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:

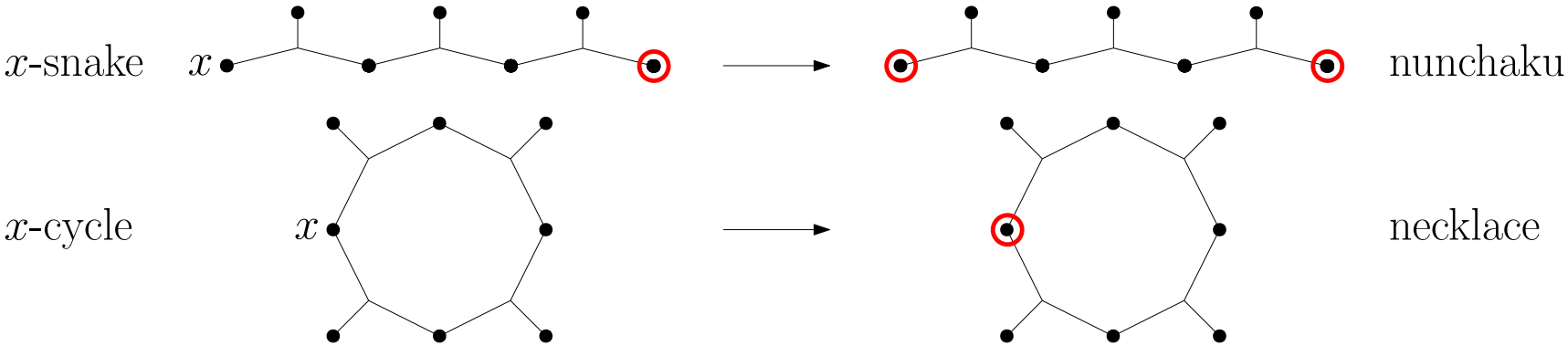


Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.

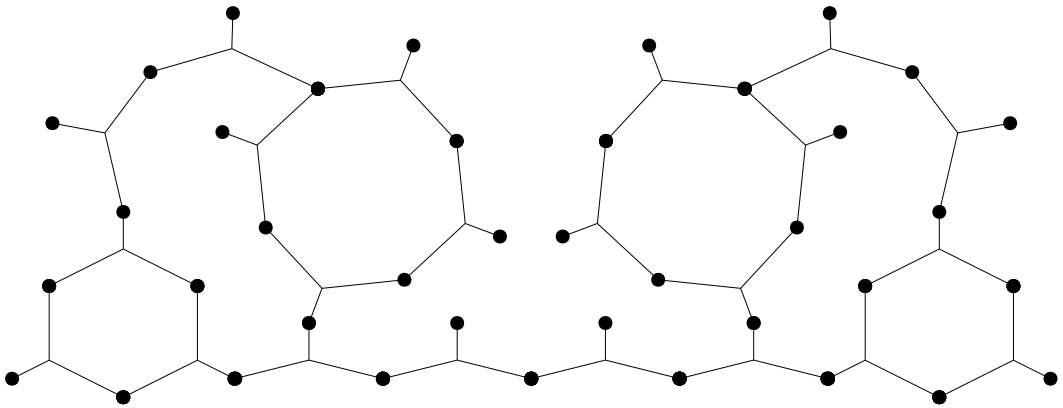
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



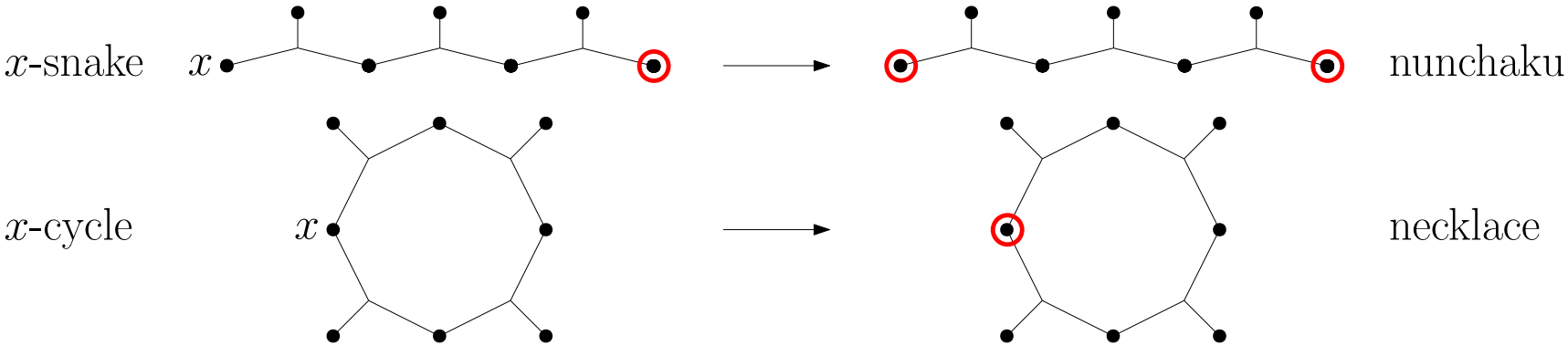
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



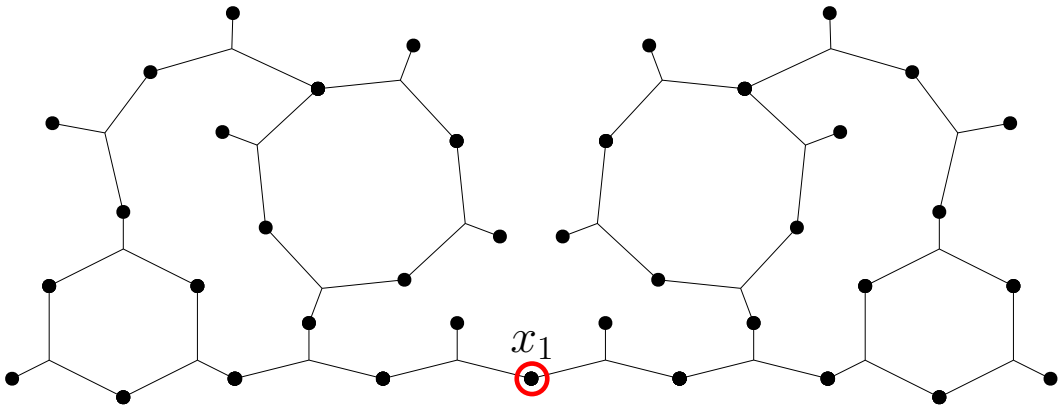
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



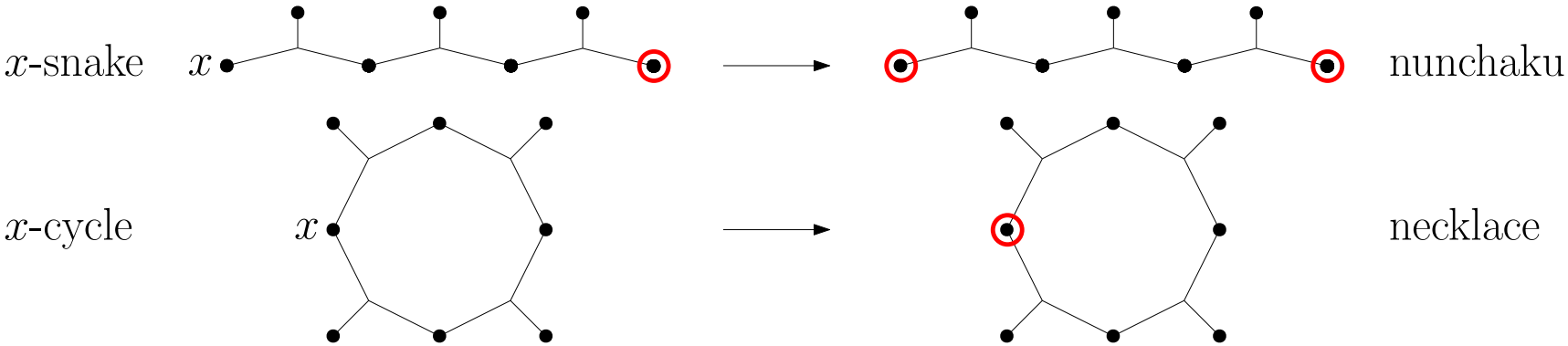
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



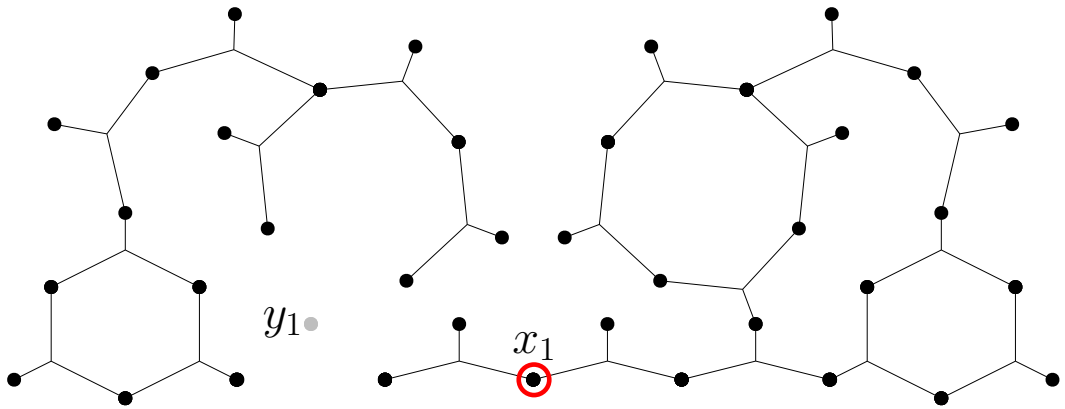
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



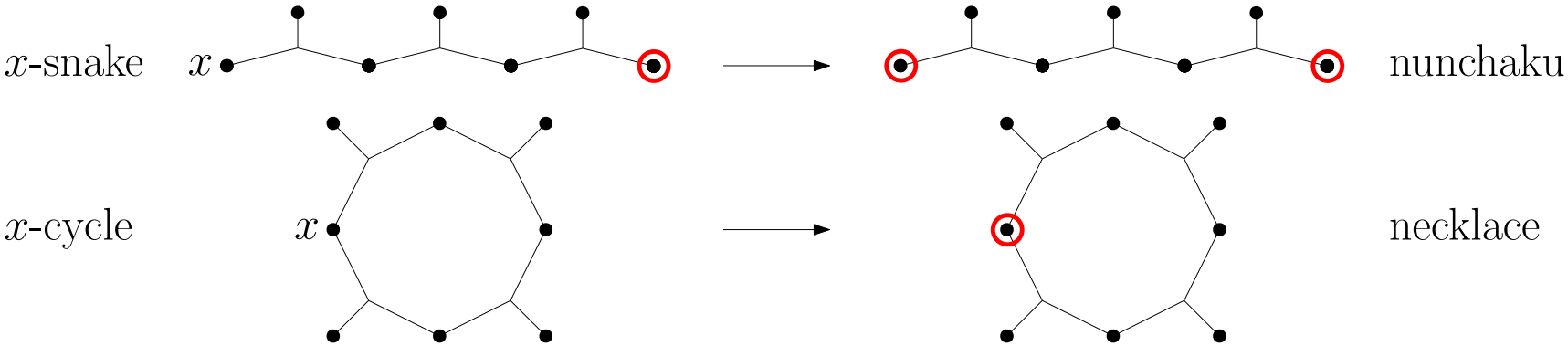
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



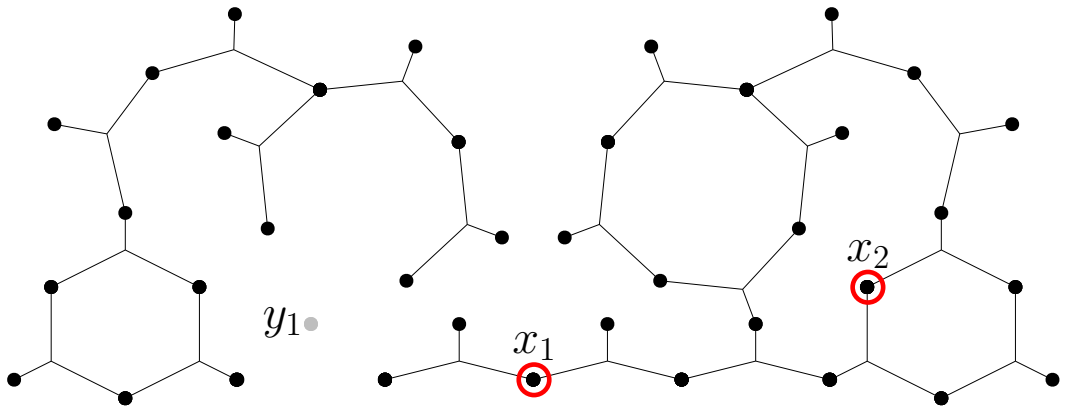
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



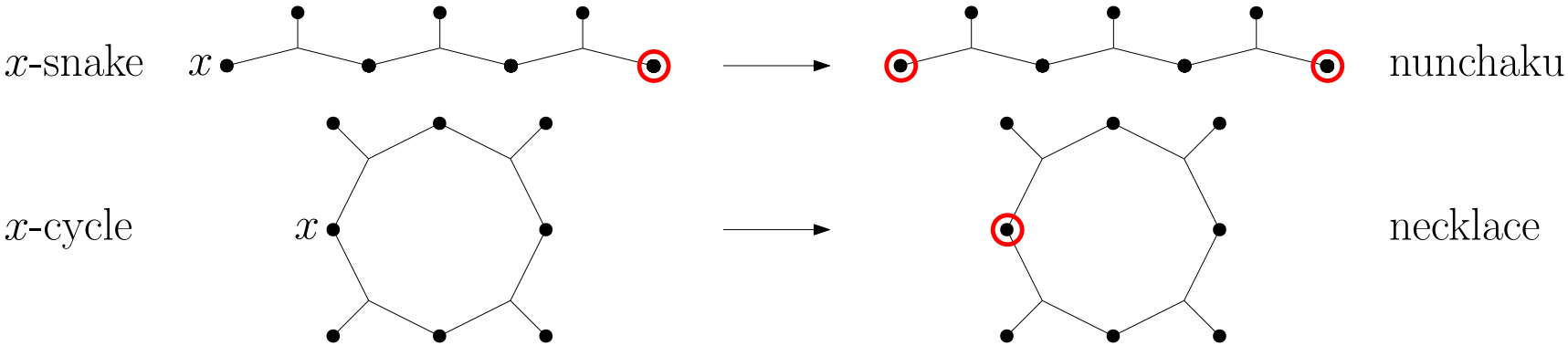
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



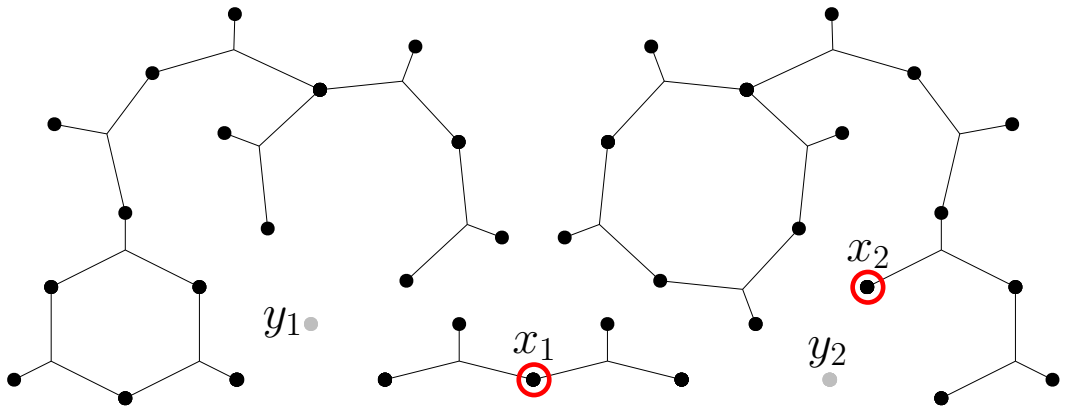
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



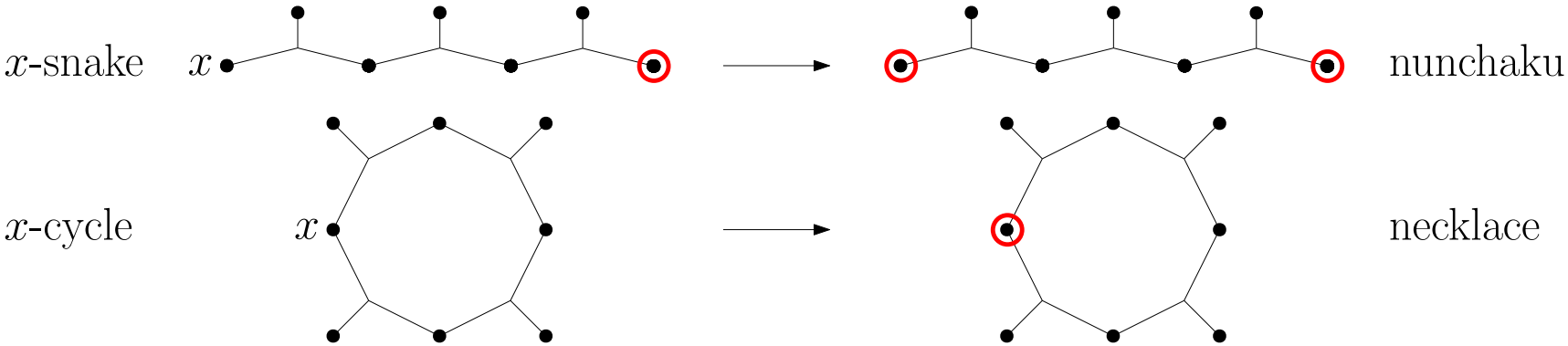
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



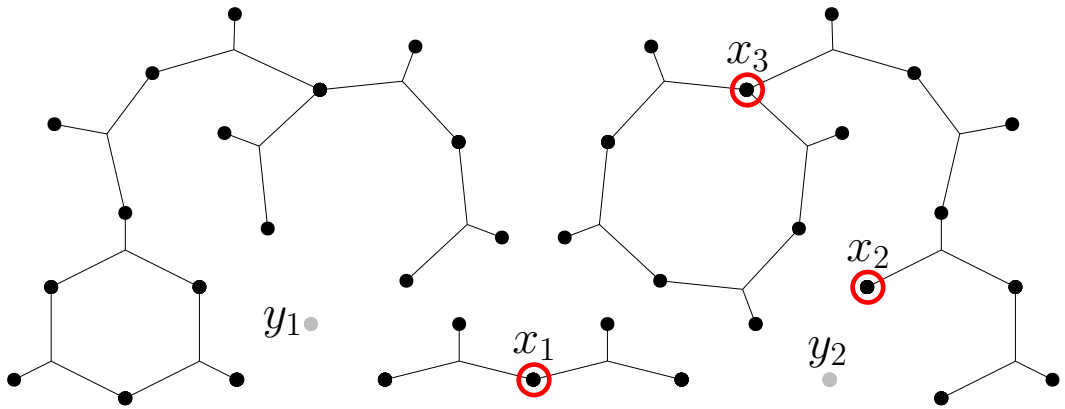
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



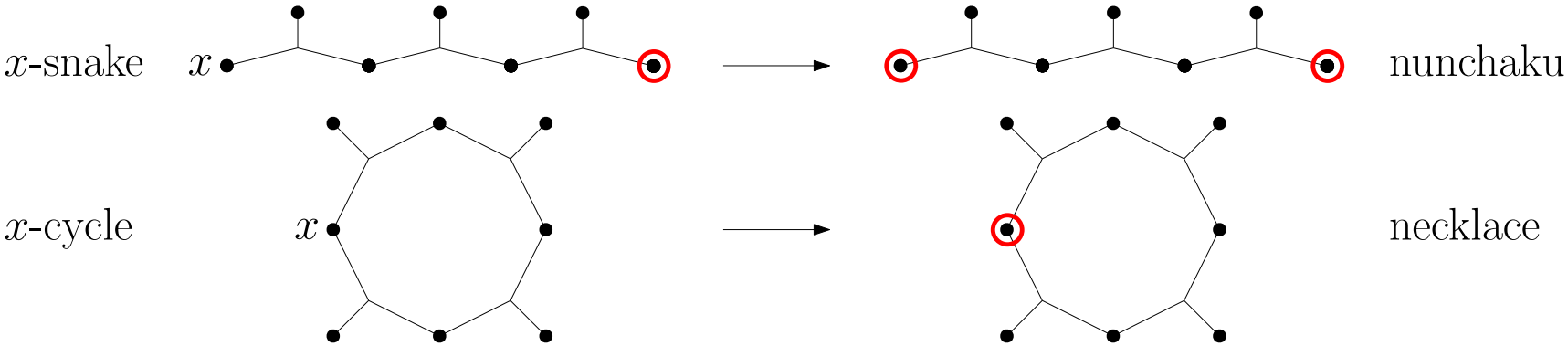
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



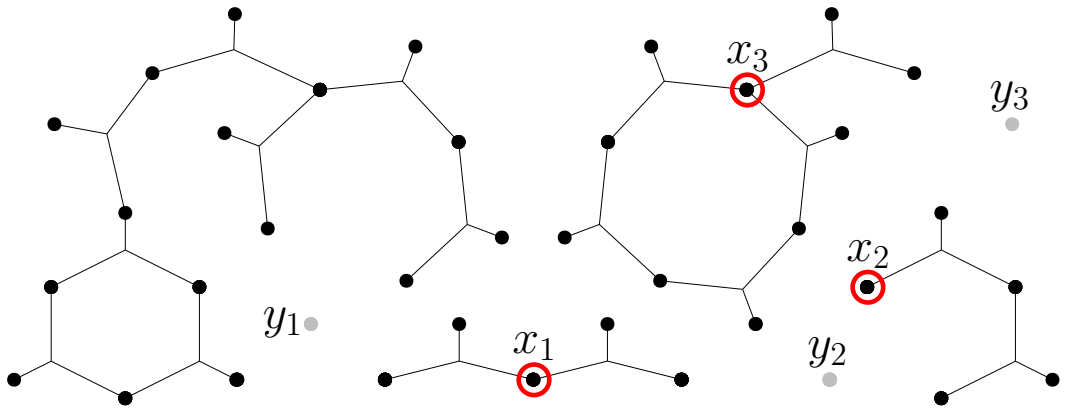
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



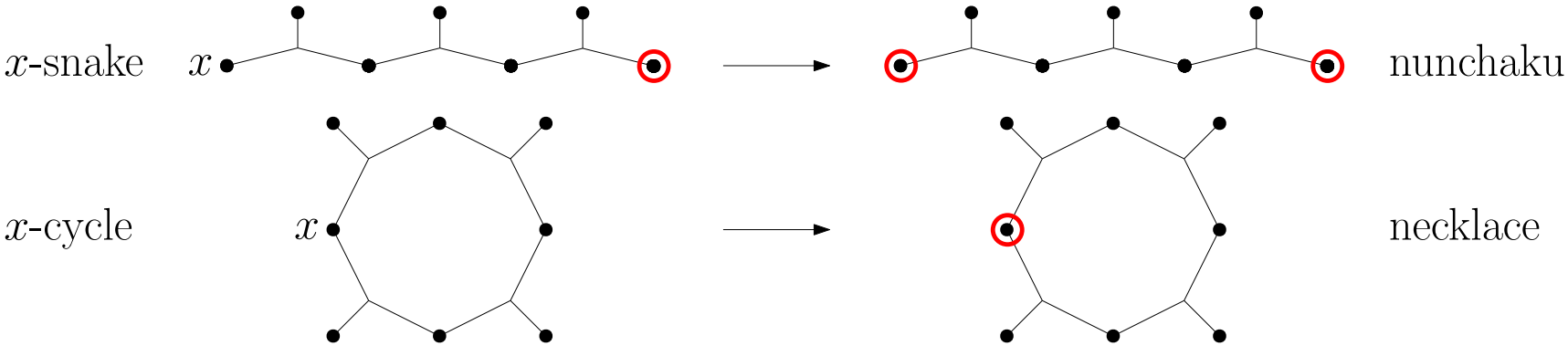
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



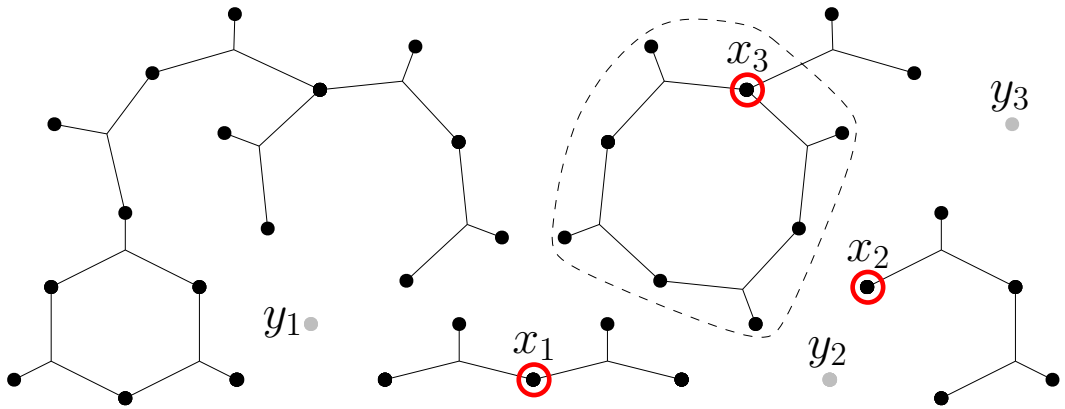
Main structural result

If Breaker fails to destroy a $\mathcal{D}_0$-danger, a *nunchaku*/*necklace* appears:



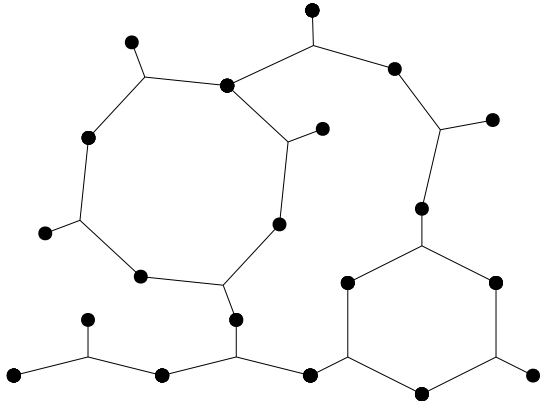
Corollary [G., Gravier, Sivignon (2022)]

Let H be a 3-uniform marked hypergraph. Then H is a Maker win if and only if Maker can guarantee that a nunchaku or necklace appears after at most three rounds of play on H.



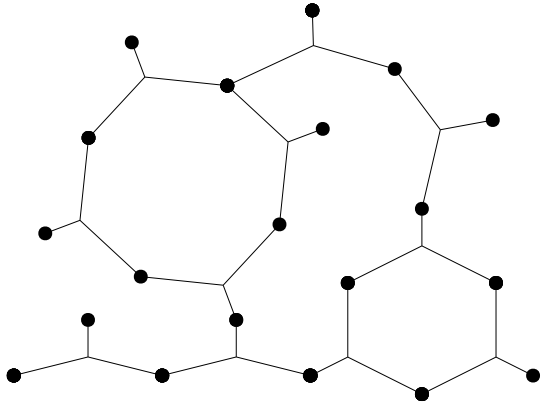
Main structural result

$\mathcal{D}_0^{**}$-dangers are **unions of unions** of chains and cycles.

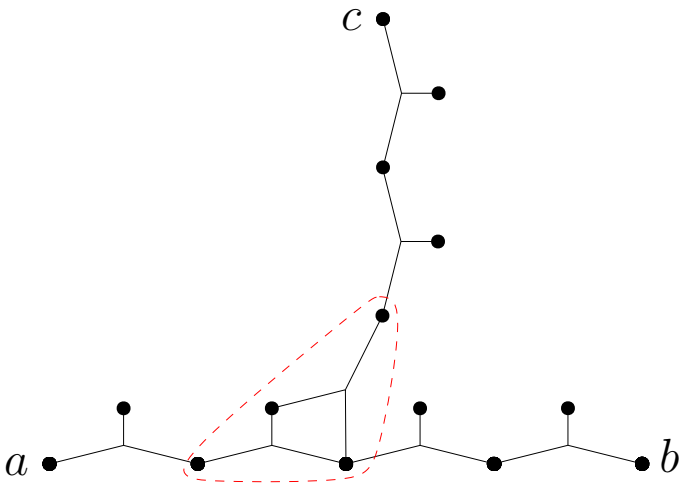


Main structural result

$\mathcal{D}_0^{**}$-dangers are **unions of unions** of chains and cycles.



Non-linearity makes the proof more difficult...



The Maker-Breaker game on hypergraphs of rank 3

1. Introduction
2. Structural result
3. Algorithmic result
4. Conclusion

Previous results

MakerBreaker decision problem

Input: a (marked) hypergraph H.
Output: YES iff H is a Maker win.

Previous results

MakerBreaker decision problem

Input: a (marked) hypergraph H.
Output: YES iff H is a Maker win.

Theorem [Schaefer, 1978]

MakerBreaker is PSPACE-complete on hypergraphs of rank 11.

Previous results

MakerBreaker decision problem

Input: a (marked) hypergraph H.
Output: YES iff H is a Maker win.

Theorem [Schaefer, 1978]

MakerBreaker is PSPACE-complete on hypergraphs of rank 11.

Theorem [Rahman & Watson, 2021]

MakerBreaker is PSPACE-complete on hypergraphs of rank 6.

Previous results

MakerBreaker decision problem

Input: a (marked) hypergraph H.
Output: YES iff H is a Maker win.

Theorem [Schaefer, 1978]

MakerBreaker is PSPACE-complete on hypergraphs of rank 11.

Theorem [Rahman & Watson, 2021]

MakerBreaker is PSPACE-complete on hypergraphs of rank 6.

Theorem [Kutz, 2004]

MakerBreaker is in polynomial time on **linear** hypergraphs of rank 3.

Previous results

MakerBreaker decision problem

Input: a (marked) hypergraph H.
Output: YES iff H is a Maker win.

Theorem [Schaefer, 1978]

MakerBreaker is PSPACE-complete on hypergraphs of rank 11.

Theorem [Rahman & Watson, 2021]

MakerBreaker is PSPACE-complete on hypergraphs of rank 6.

Theorem [Kutz, 2004]

MakerBreaker is in polynomial time on **linear** hypergraphs of rank 3.

Conjecture [Rahman & Watson, 2020]

MakerBreaker is in polynomial time on **all** hypergraphs of rank 3.

Reduction to the chain existence problem

- For a general hypergraph H on n vertices:

Maker wins $\iff$ $\exists\, x_1, \forall\, y_1, \exists\, x_2, \forall\, y_2, \exists\, x_3, \forall\, y_3, \exists\, x_4, \forall\, y_4, \ldots, \exists\, x_{\frac{n}{2}}, \forall\, y_{\frac{n}{2}}$,

$\{x_1, \ldots, x_{\frac{n}{2}}\}$ contains an edge of H.

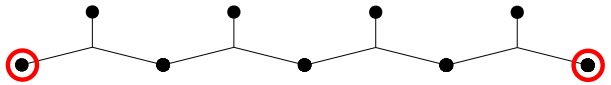
Reduction to the chain existence problem

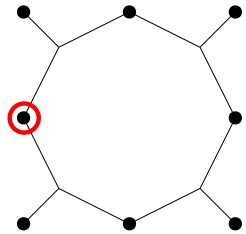
- For a general hypergraph H on n vertices:

Maker wins $\iff$ $\exists\, x_1, \forall\, y_1, \exists\, x_2, \forall\, y_2, \exists\, x_3, \forall\, y_3, \exists\, x_4, \forall\, y_4, \ldots, \exists\, x_{\frac{n}{2}}, \forall\, y_{\frac{n}{2}}$,
$\{x_1, \ldots, x_{\frac{n}{2}}\}$ contains an edge of H.

- For a **3-uniform** marked hypergraph H:

Maker wins $\iff$ $\exists\, x_1, \forall\, y_1, \exists\, x_2, \forall\, y_2, \exists\, x_3, \forall\, y_3$,
there is a nunchaku/necklace in $H^{+x_1-y_1+x_2-y_2+x_3-y_3}$.





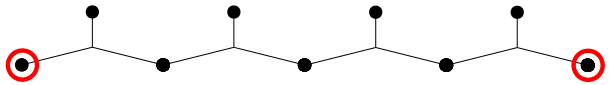
Reduction to the chain existence problem

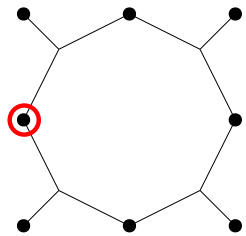
- For a general hypergraph H on n vertices:

Maker wins $\iff$ $\exists\, x_1, \forall\, y_1, \exists\, x_2, \forall\, y_2, \exists\, x_3, \forall\, y_3, \exists\, x_4, \forall\, y_4, \ldots, \exists\, x_{\frac{n}{2}}, \forall\, y_{\frac{n}{2}}$,
$\{x_1, \ldots, x_{\frac{n}{2}}\}$ contains an edge of H.

- For a **3-uniform** marked hypergraph H:

Maker wins $\iff$ $\exists\, x_1, \forall\, y_1, \exists\, x_2, \forall\, y_2, \exists\, x_3, \forall\, y_3$,
there is a nunchaku/necklace in $H^{+x_1-y_1+x_2-y_2+x_3-y_3}$.





Corollary

MakerBreaker on hypergraphs of rank 3 reduces to LinearCon-3.

LinearCon-3 decision problem

Input: A 3-uniform hypergraph H and two vertices x, y of H.
Output: YES iff there exists an xy-chain in H.

Linear connected component: definition

- We are going to compute the associated **connected components**.

We fix $x^* \in V(H)$.

Linear connected component: definition

- We are going to compute the associated **connected components**.

We fix $x^* \in V(H)$.

Definition

The *linear connected component* of x^* in H is the set $LCC_H(x^*)$ of all $x \in V(H)$ such that there exists an x^*x-chain in H.
In practice, this term will refer to the induced subhypergraph $H[LCC_H(x^*)]$.

Linear connected component: exploration

Goal: an algorithm that computes the linear connected component of x^*.

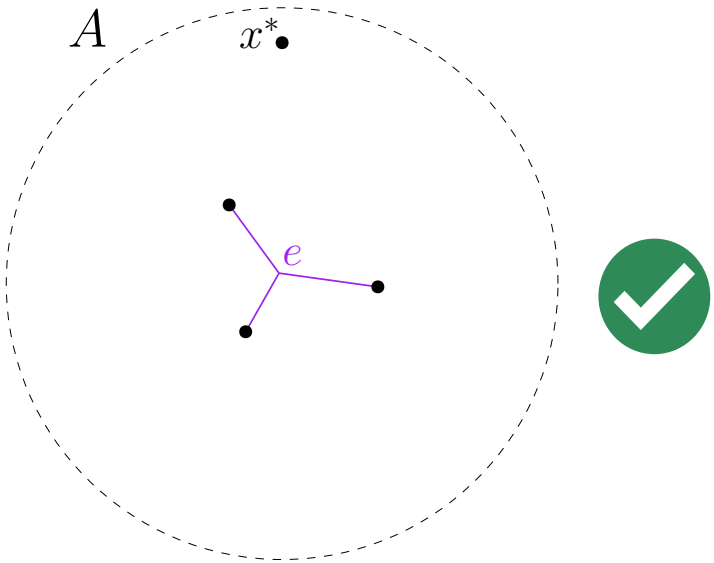
- **Exploration on the edges.**
- Initialization = subhypergraph reduced to the vertex x^*.
- Each examined edge is either accepted (added to the subhypergraph we are building) or rejected (temporarily).

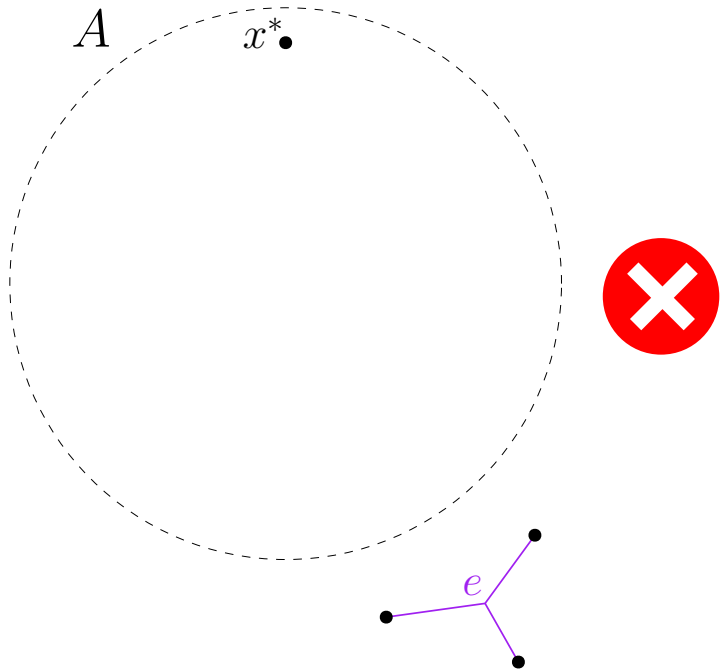
Linear connected component: exploration

Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.

Linear connected component: exploration

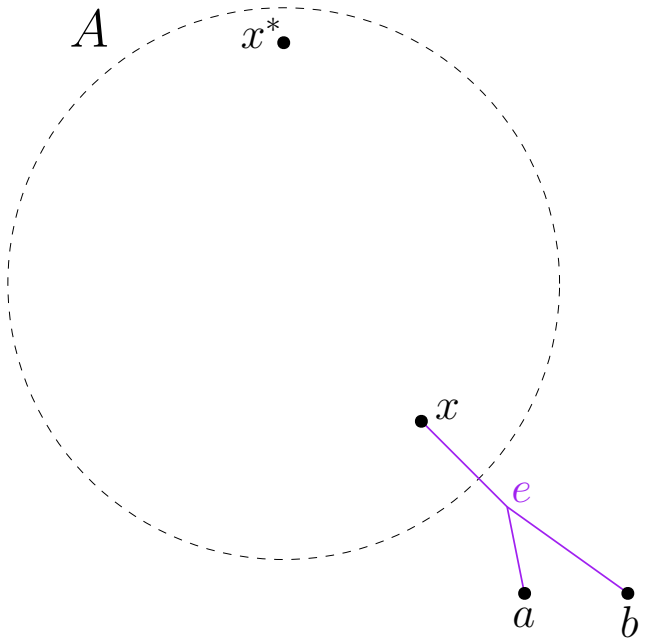
Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.





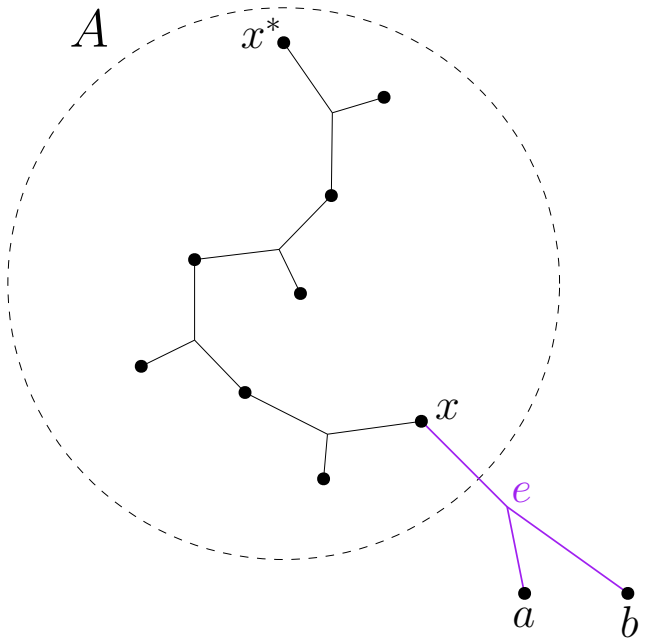
Linear connected component: exploration

Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.



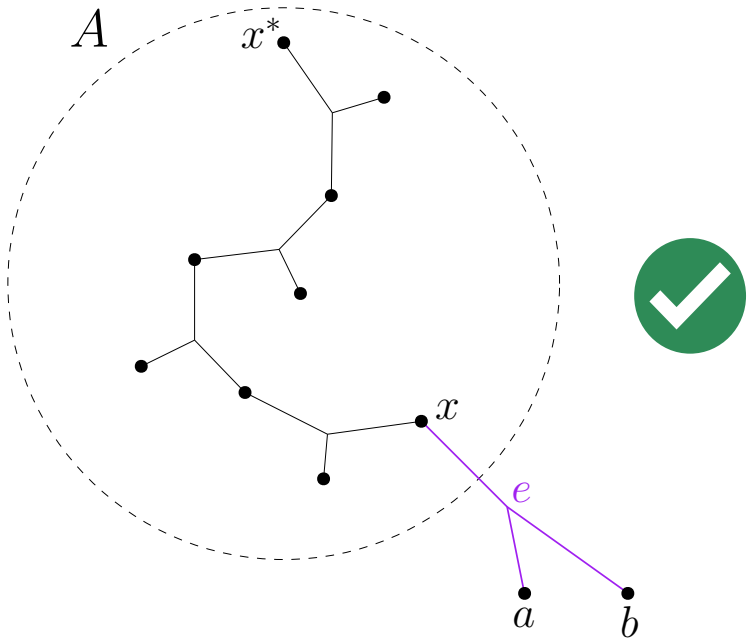
Linear connected component: exploration

Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.



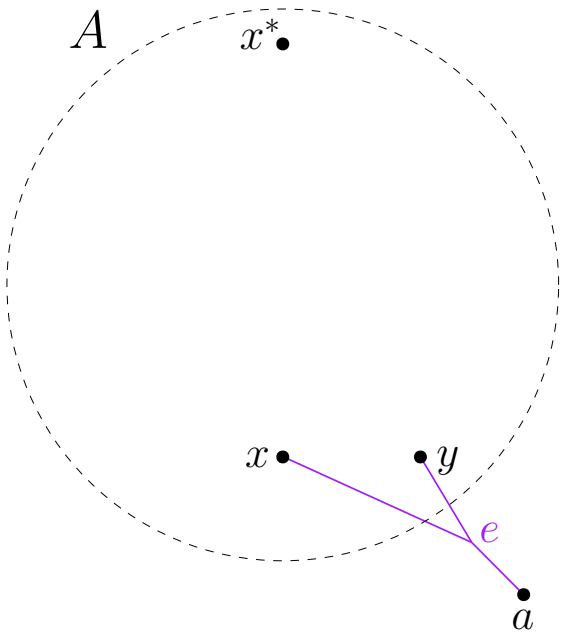
Linear connected component: exploration

Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.



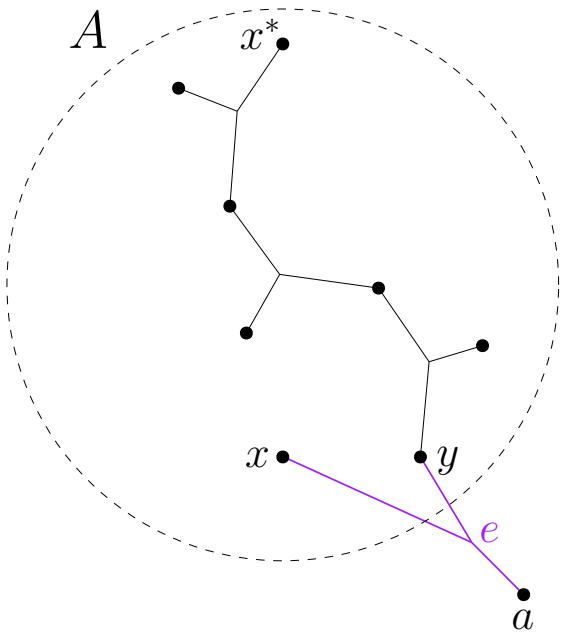
Linear connected component: exploration

Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.



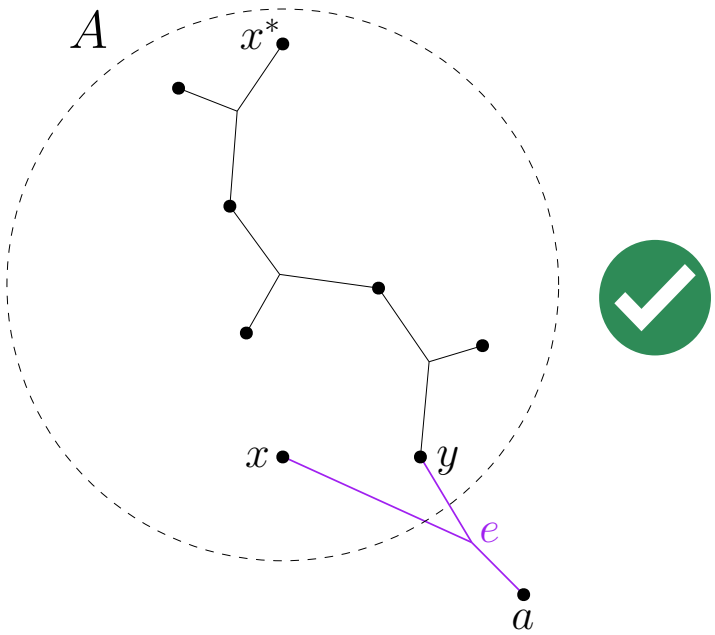
Linear connected component: exploration

Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.



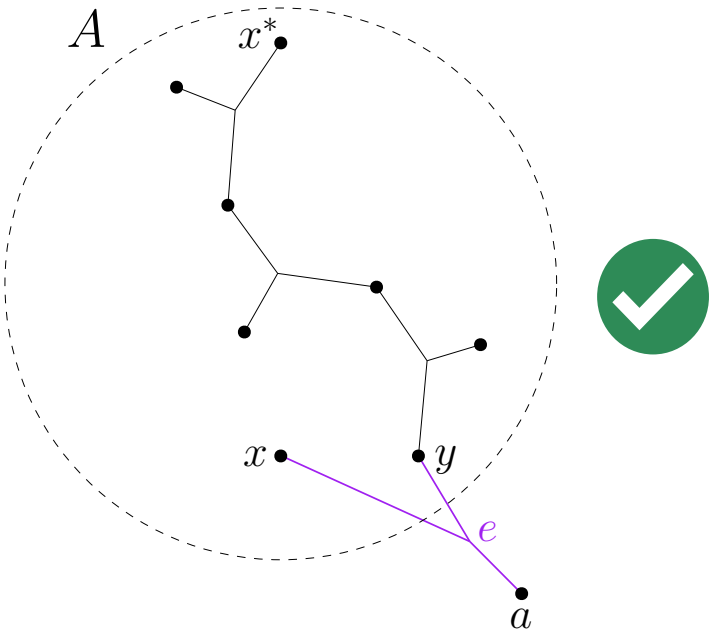
Linear connected component: exploration

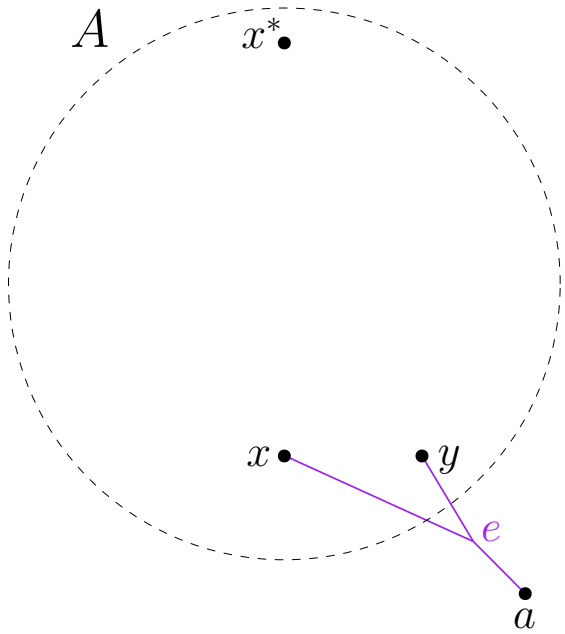
Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.



Linear connected component: exploration

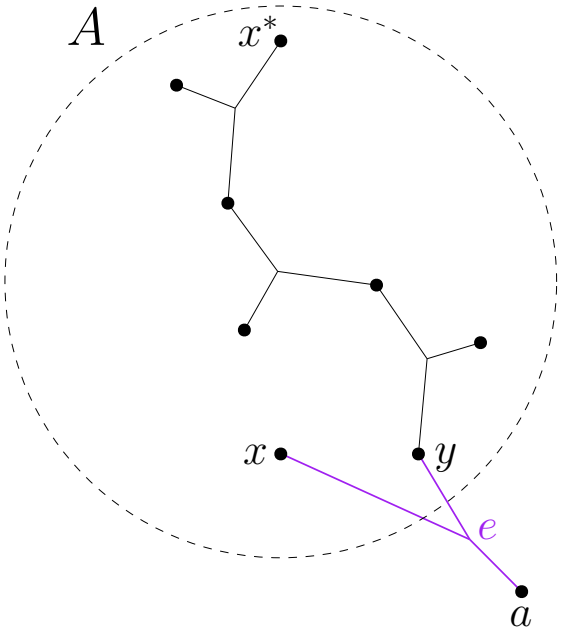
Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.

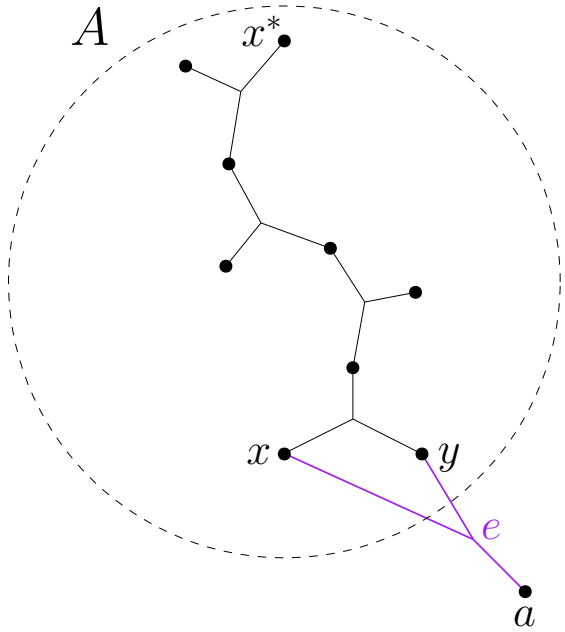




Linear connected component: exploration

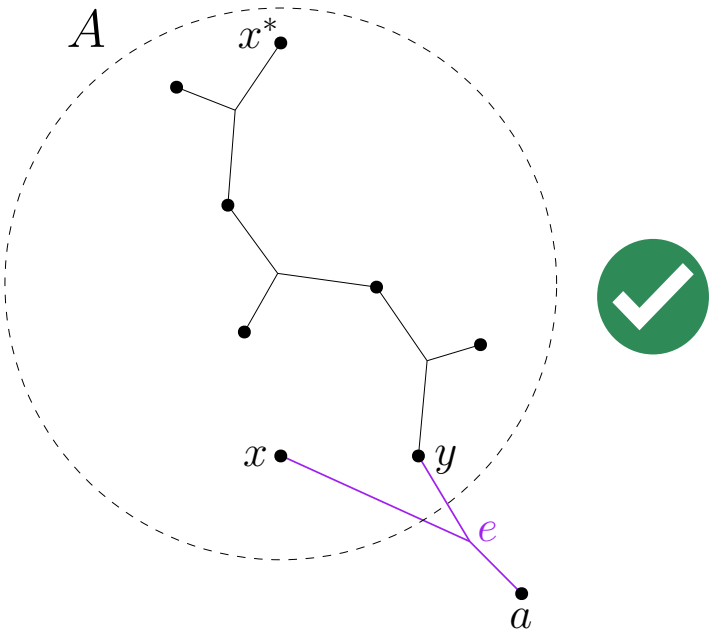
Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.

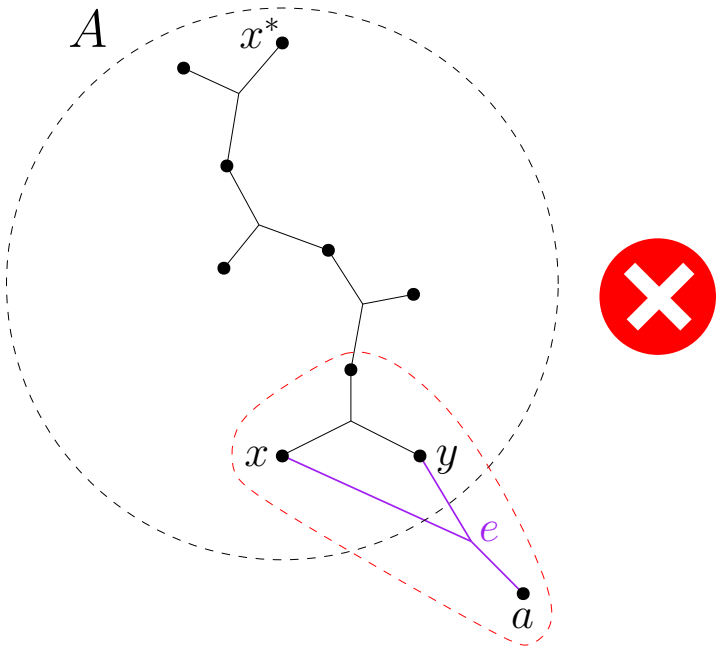




Linear connected component: exploration

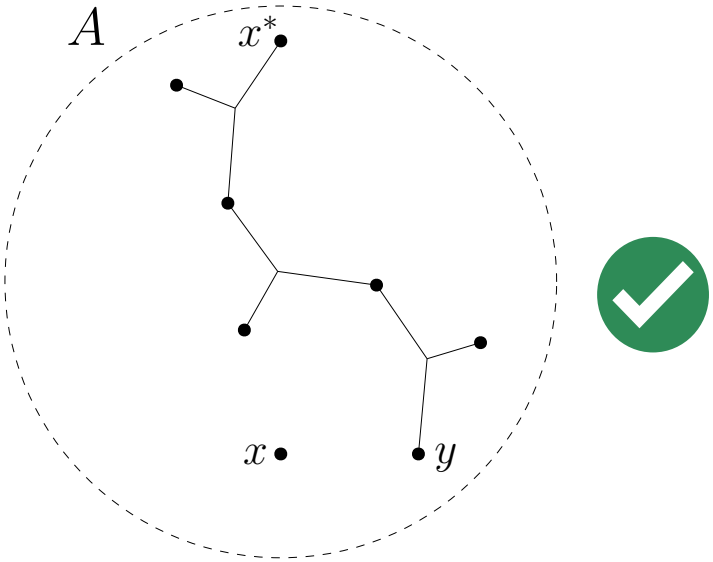
Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.

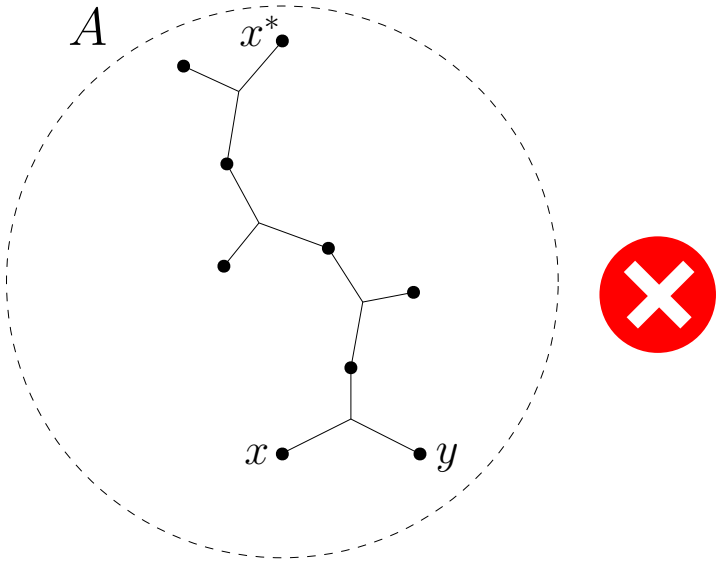




Linear connected component: exploration

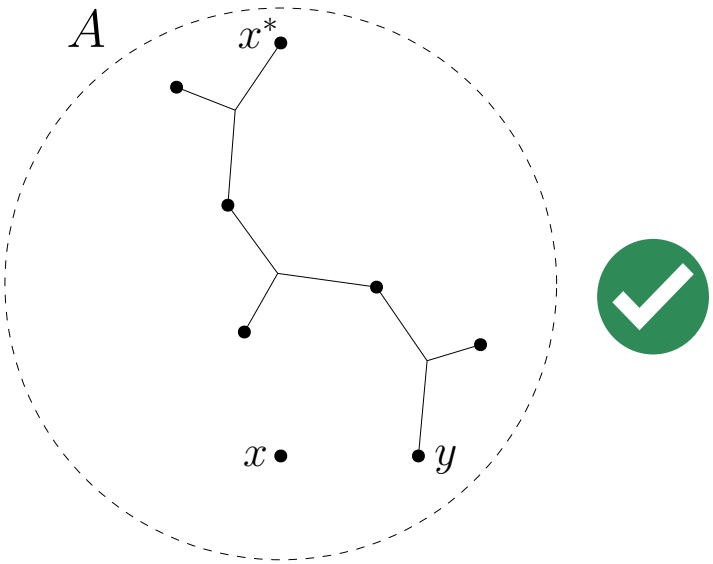
Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.

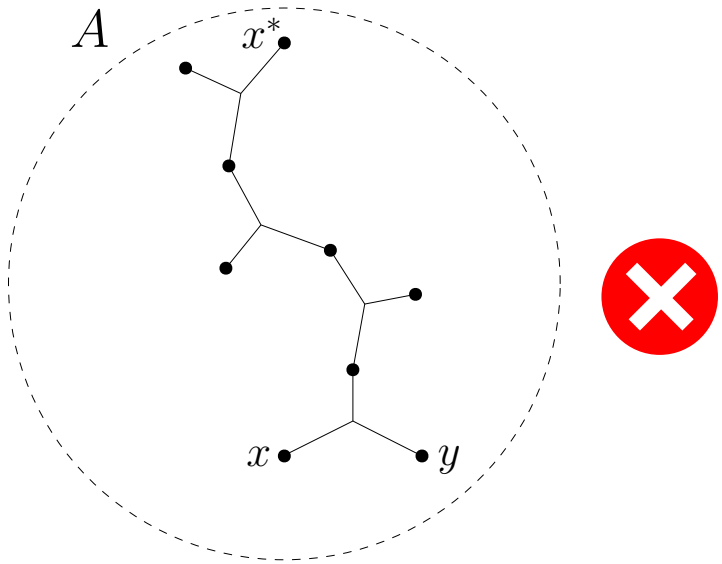




Linear connected component: exploration

Say that, during the algorithm's execution, we have rebuilt some part A of the LCC, and we encounter a new edge e.

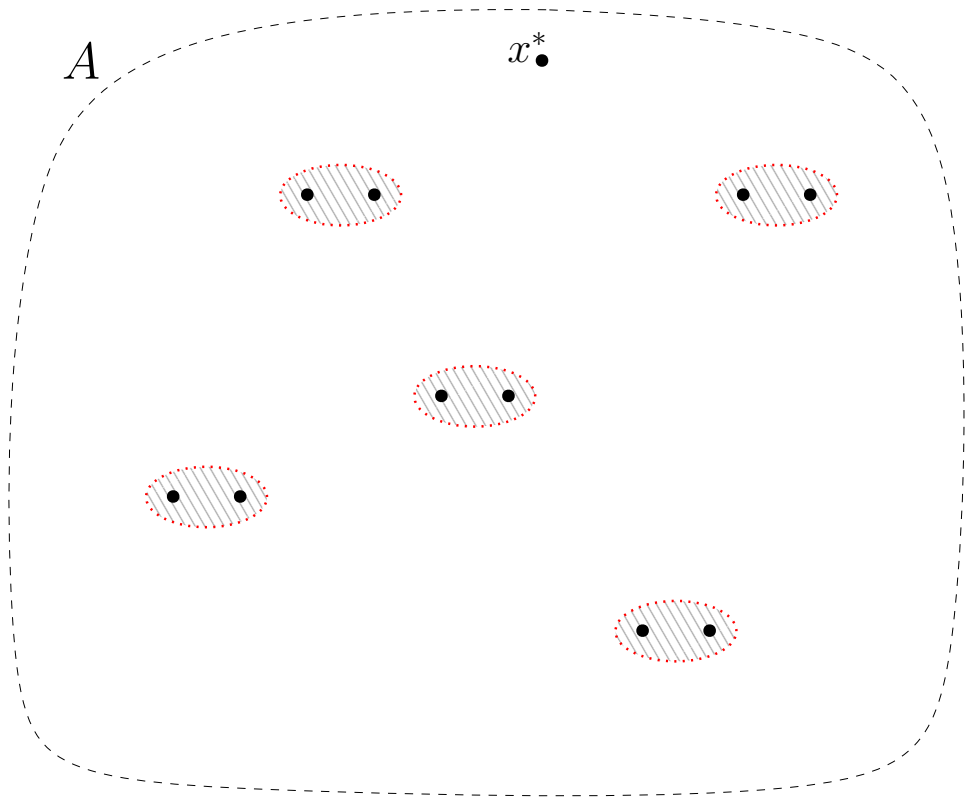




Crucial information: which are the *inseparable pairs* in A i.e. $\{x, y\}$ such that all x^*x-chains in A contain y and all x^*y-chains in A contain x.

- **The algorithm must keep this information updated.**

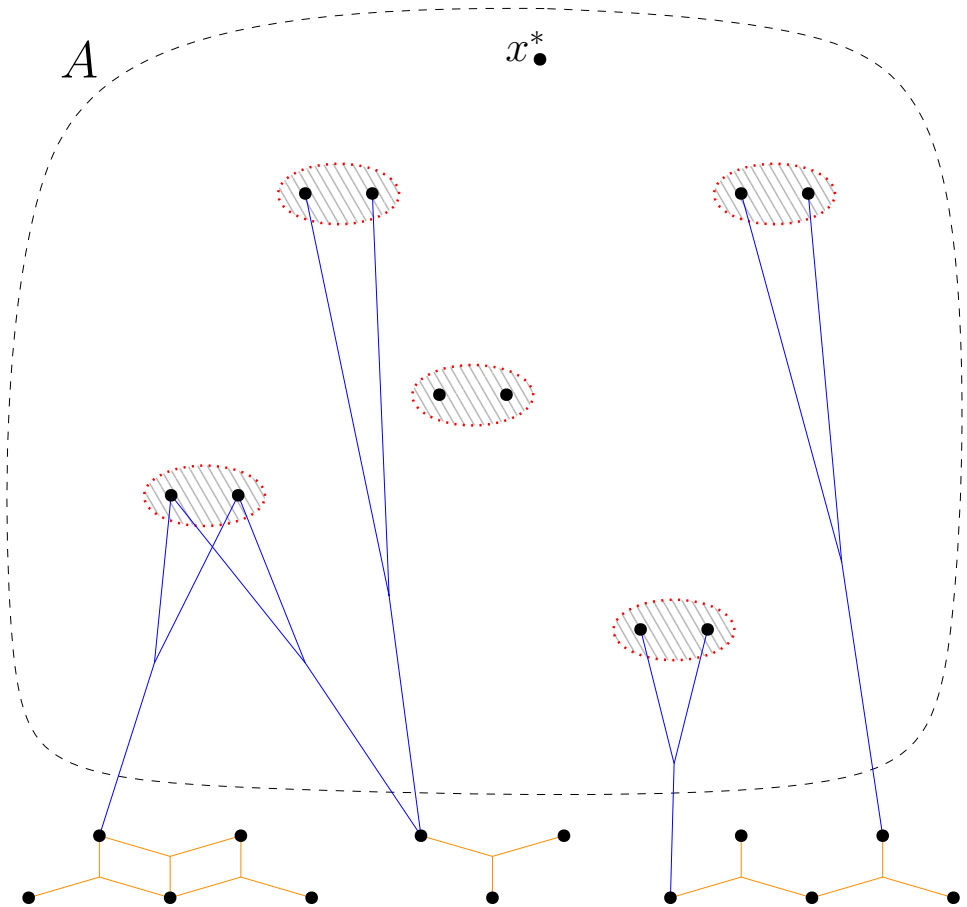
When is the search over?



When is the search over?

An edge e gets **rejected** if and only if it is either:

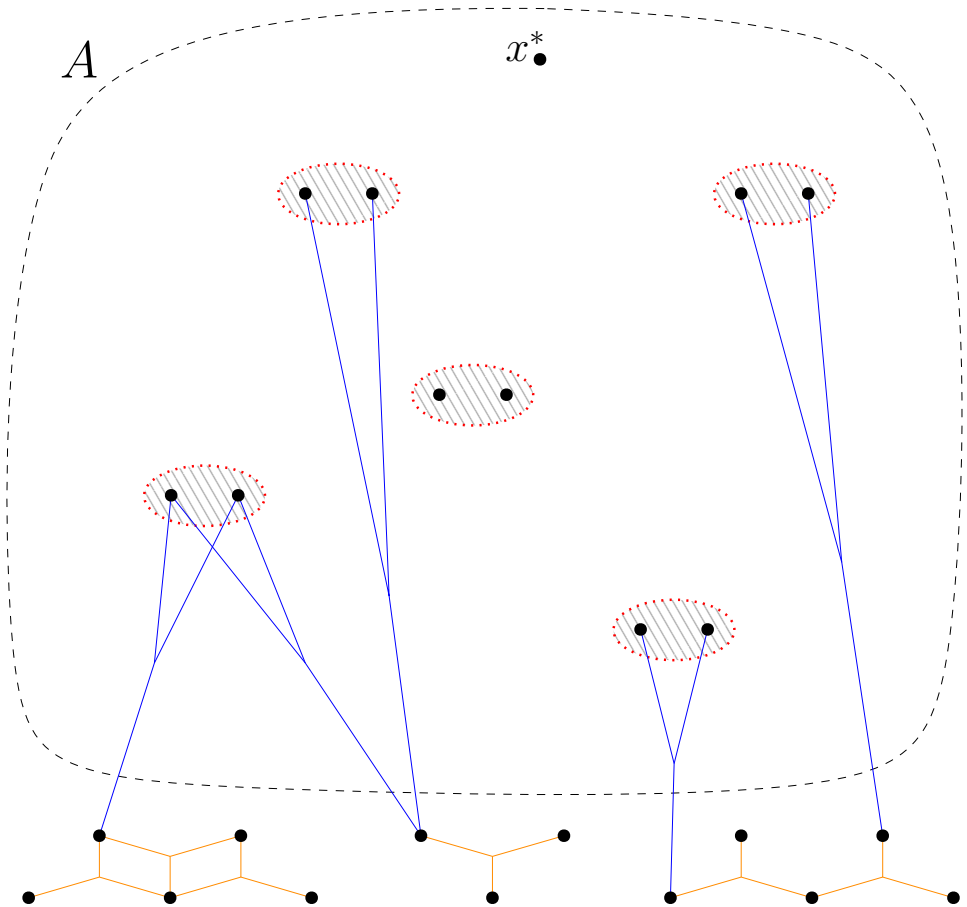
- an "exterior" edge: $e \cap V(A) = \varnothing$;
- a "cut" edge: $e \cap V(A)$ is an inseparable pair.



When is the search over?

An edge e gets **rejected** if and only if it is either:

- an "exterior" edge: $e \cap V(A) = \varnothing$;
- a "cut" edge: $e \cap V(A)$ is an inseparable pair.

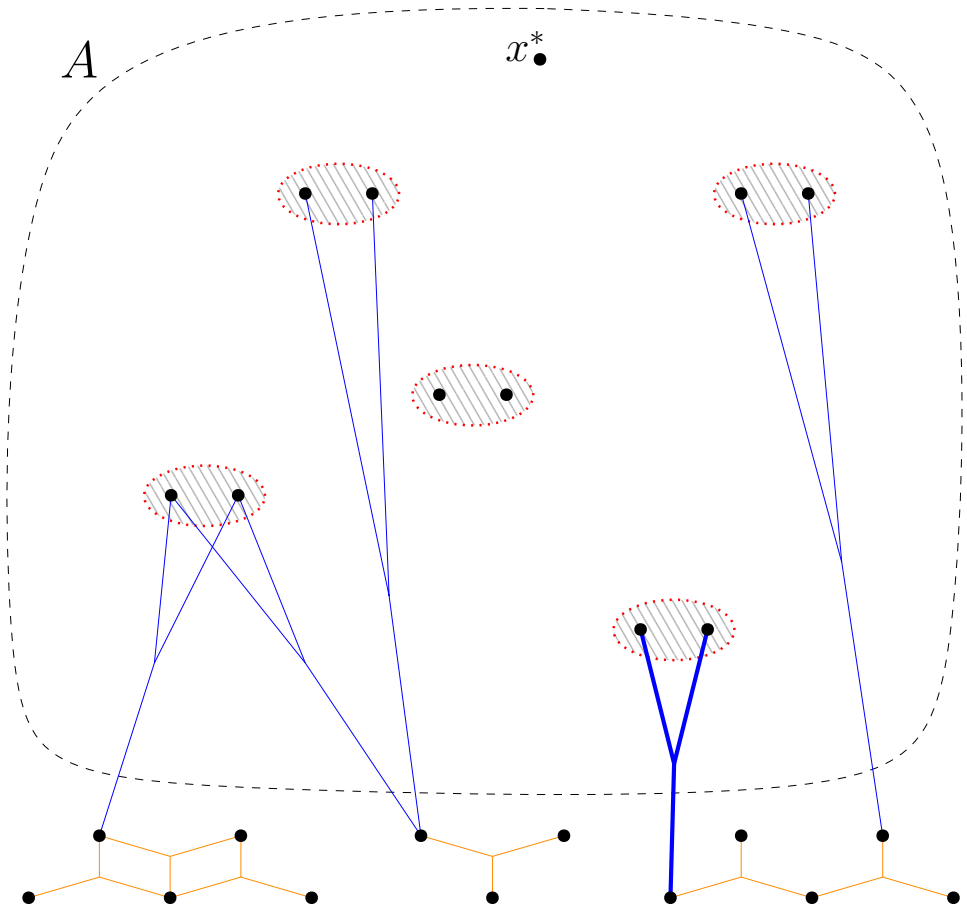


- $A = H[LCC_H(x^*)] \iff \forall\, e \in E(H) \setminus E(A)$, e is "exterior" or "cut".

When is the search over?

An edge e gets **rejected** if and only if it is either:

- an "exterior" edge: $e \cap V(A) = \varnothing$;
- a "cut" edge: $e \cap V(A)$ is an inseparable pair.

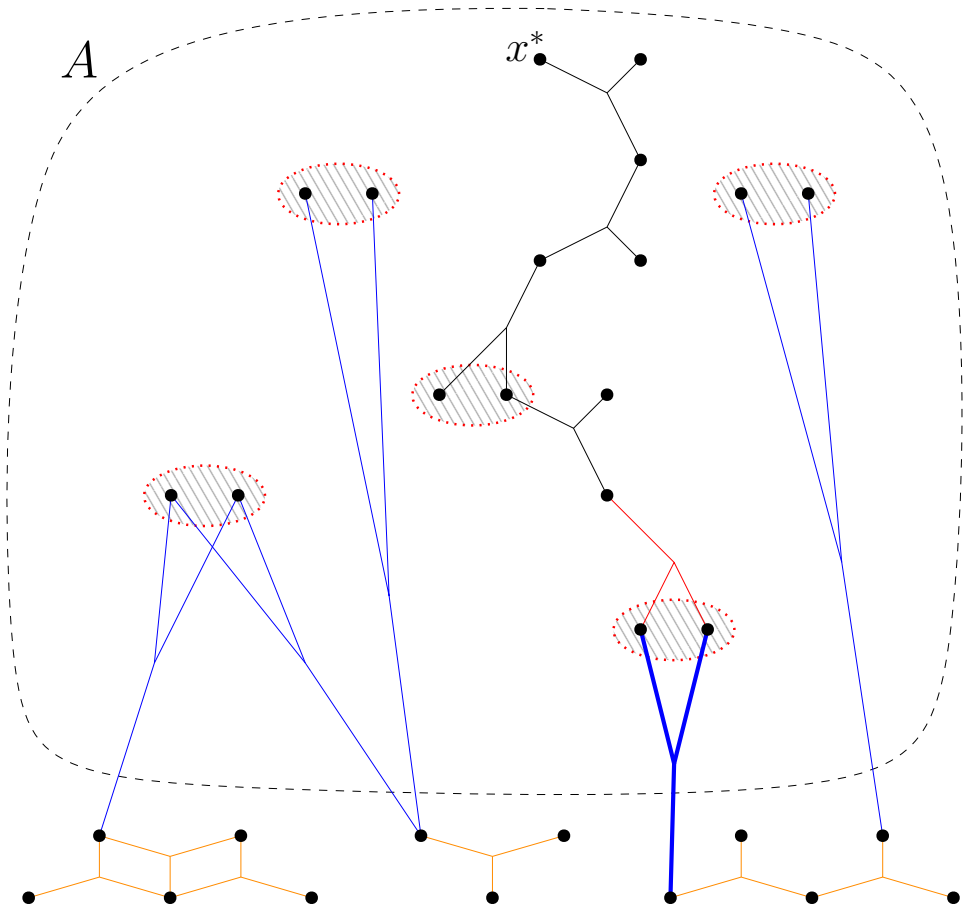


- $A = H[LCC_H(x^*)] \iff \forall\, e \in E(H) \setminus E(A)$, e is "exterior" or "cut".

When is the search over?

An edge e gets **rejected** if and only if it is either:

- an "exterior" edge: $e \cap V(A) = \varnothing$;
- a "cut" edge: $e \cap V(A)$ is an inseparable pair.

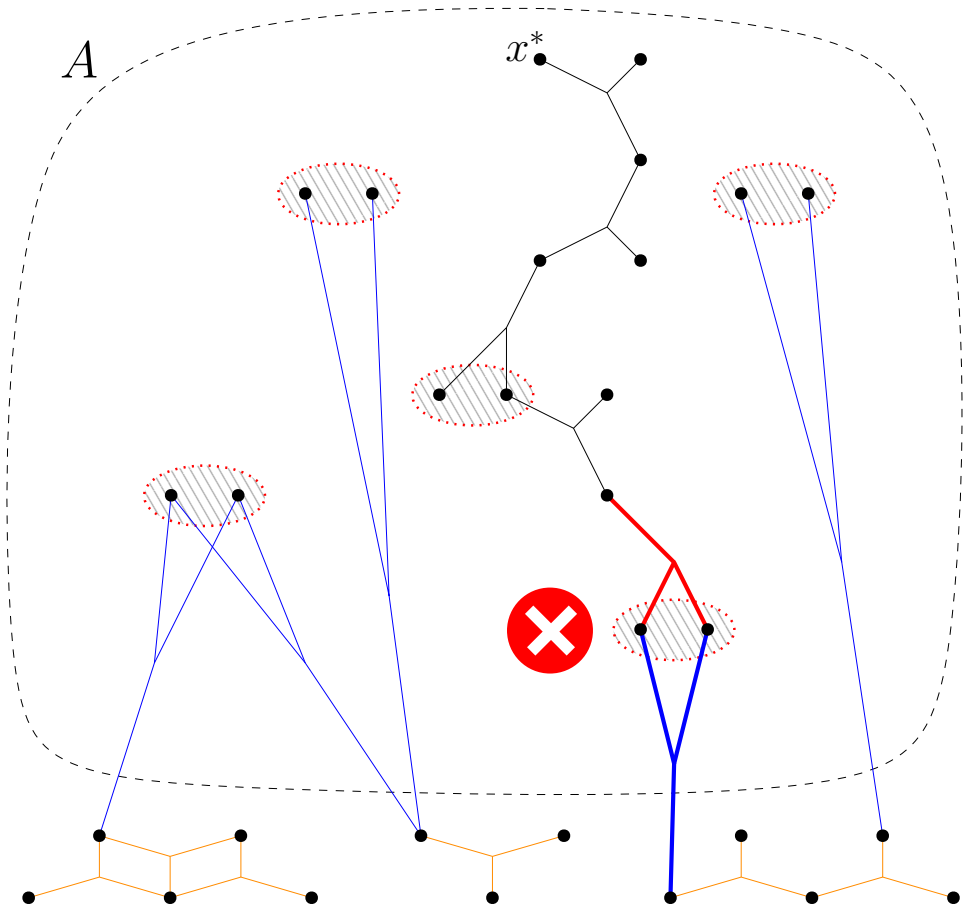


- $A = H[LCC_H(x^*)] \iff \forall\, e \in E(H) \setminus E(A)$, e is "exterior" or "cut".

When is the search over?

An edge e gets **rejected** if and only if it is either:

- an "exterior" edge: $e \cap V(A) = \varnothing$;
- a "cut" edge: $e \cap V(A)$ is an inseparable pair.



- $A = H[LCC_H(x^*)] \iff \forall\, e \in E(H) \setminus E(A)$, e is "exterior" or "cut".

The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

The idea for the algorithm

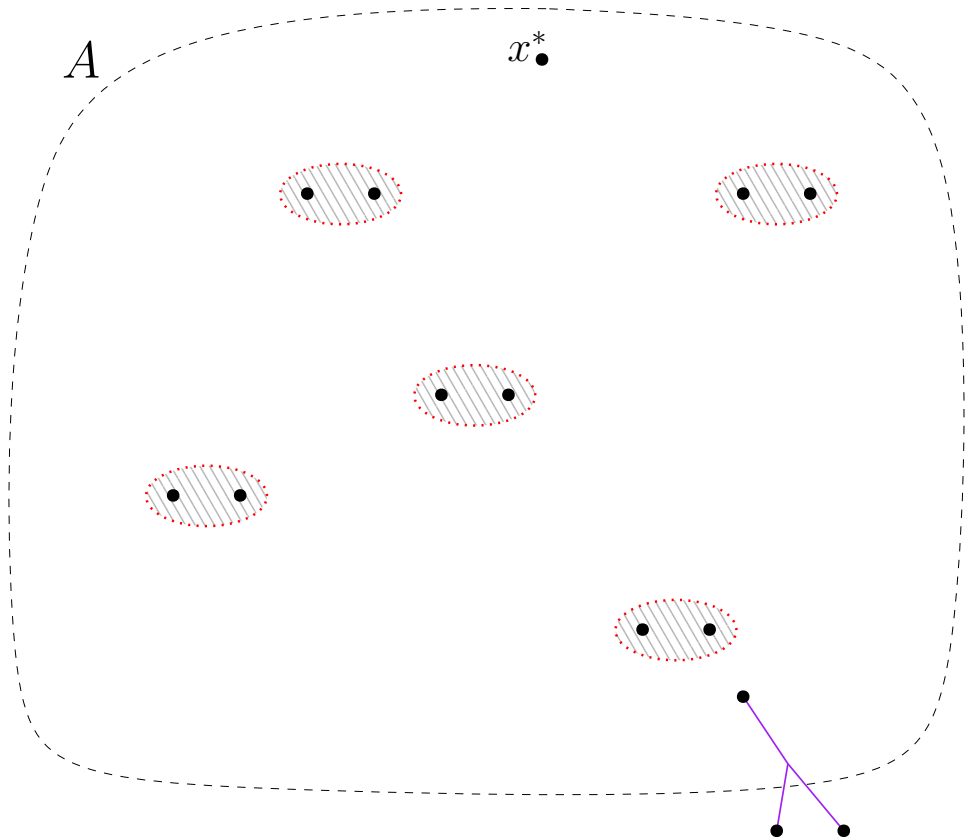
Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

- **Problem: how to update the inseparable pairs?**

The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

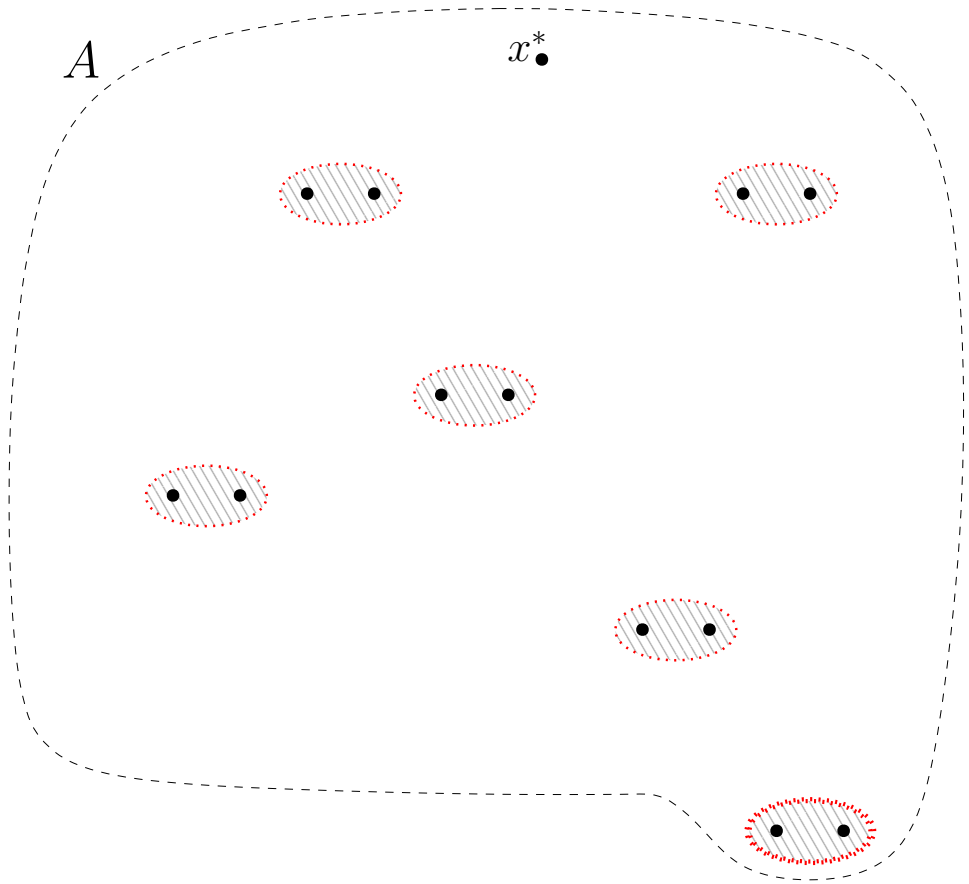
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

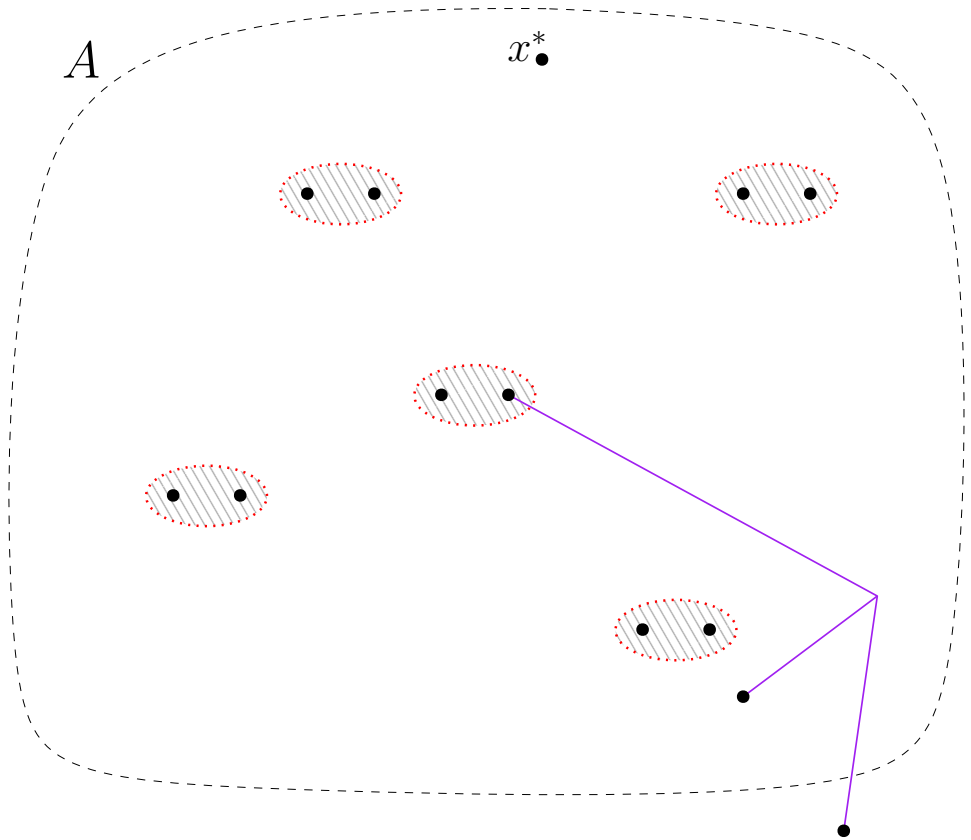
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

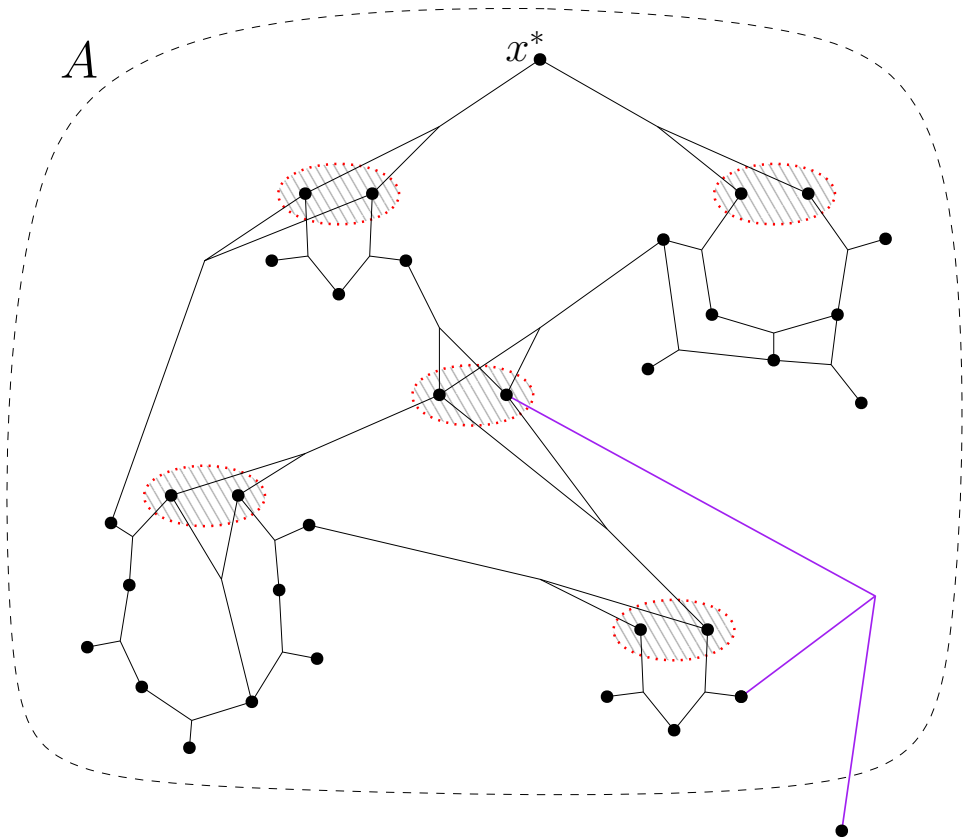
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

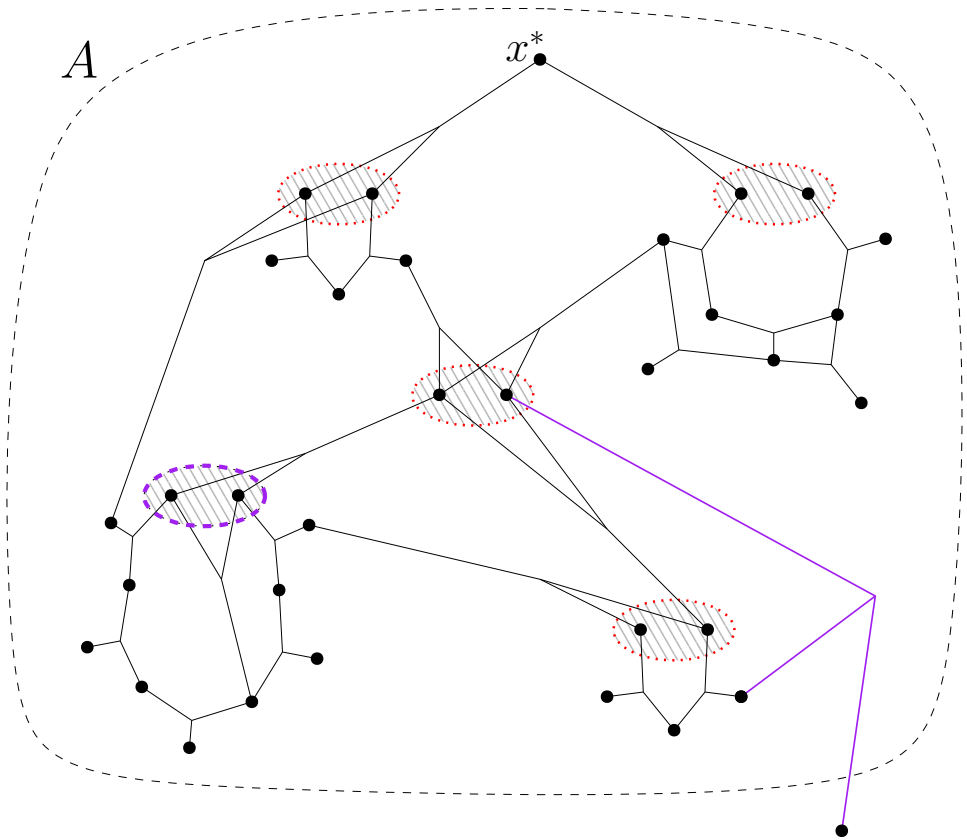
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

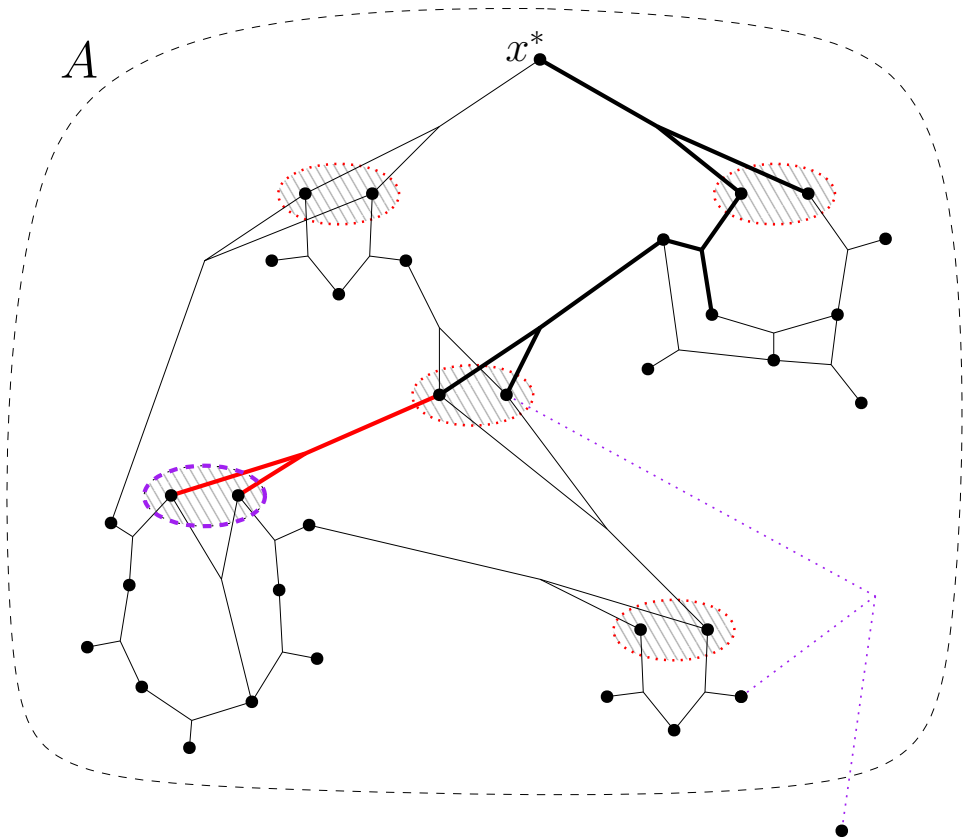
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

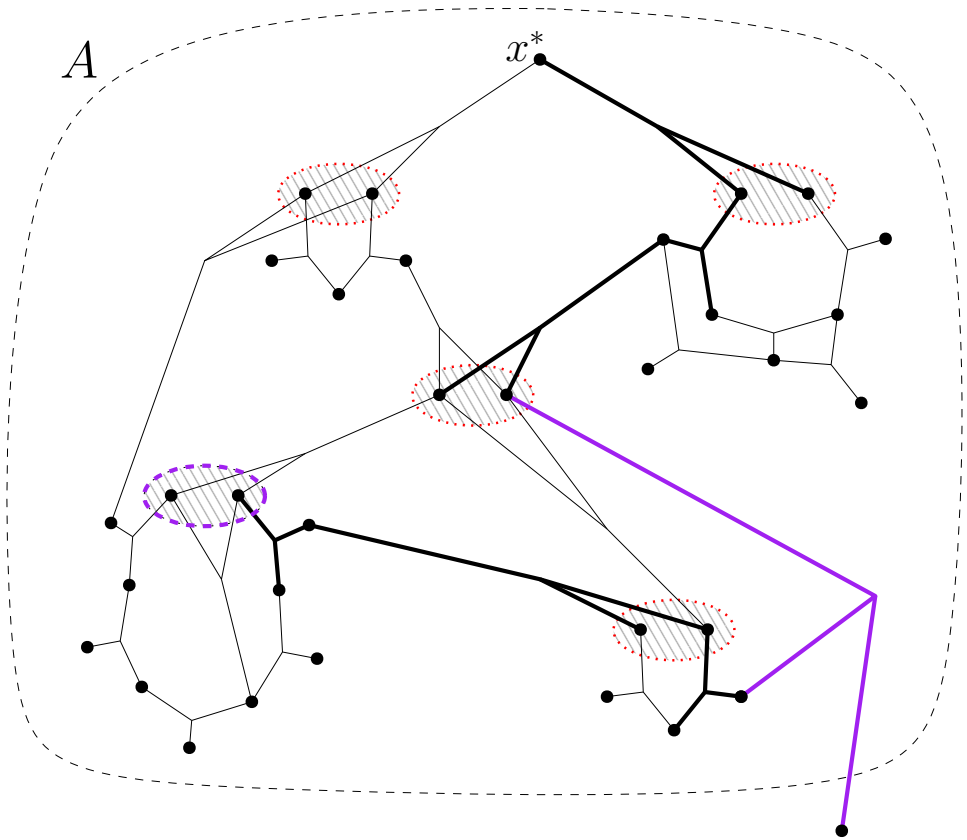
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

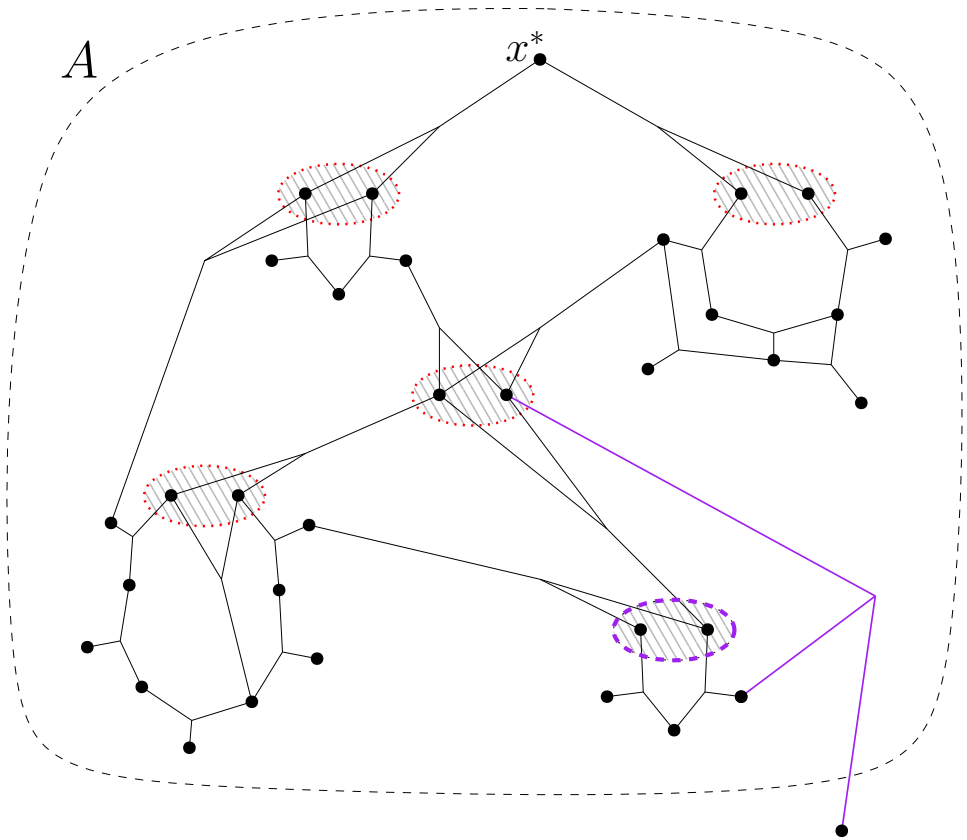
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

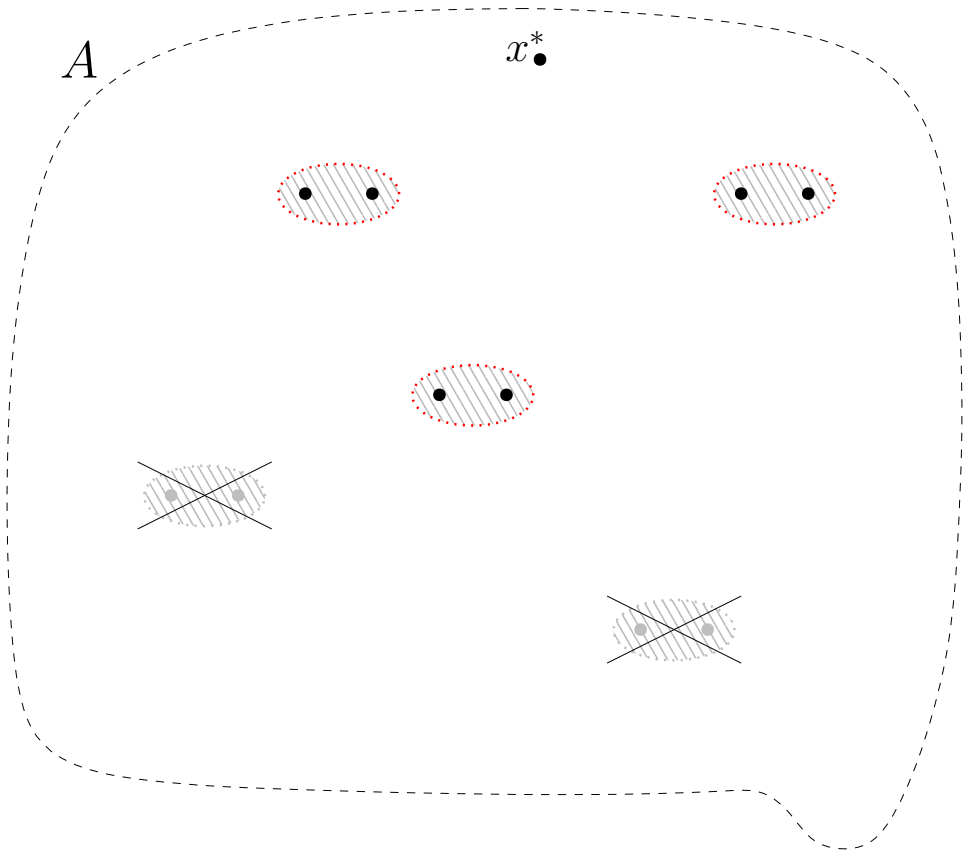
- **Problem: how to update the inseparable pairs?**



The idea for the algorithm

Algorithm: add edges that are not "exterior" or "cut", one by one, until none exist anymore at which point we are done.

- **Problem: how to update the inseparable pairs?**



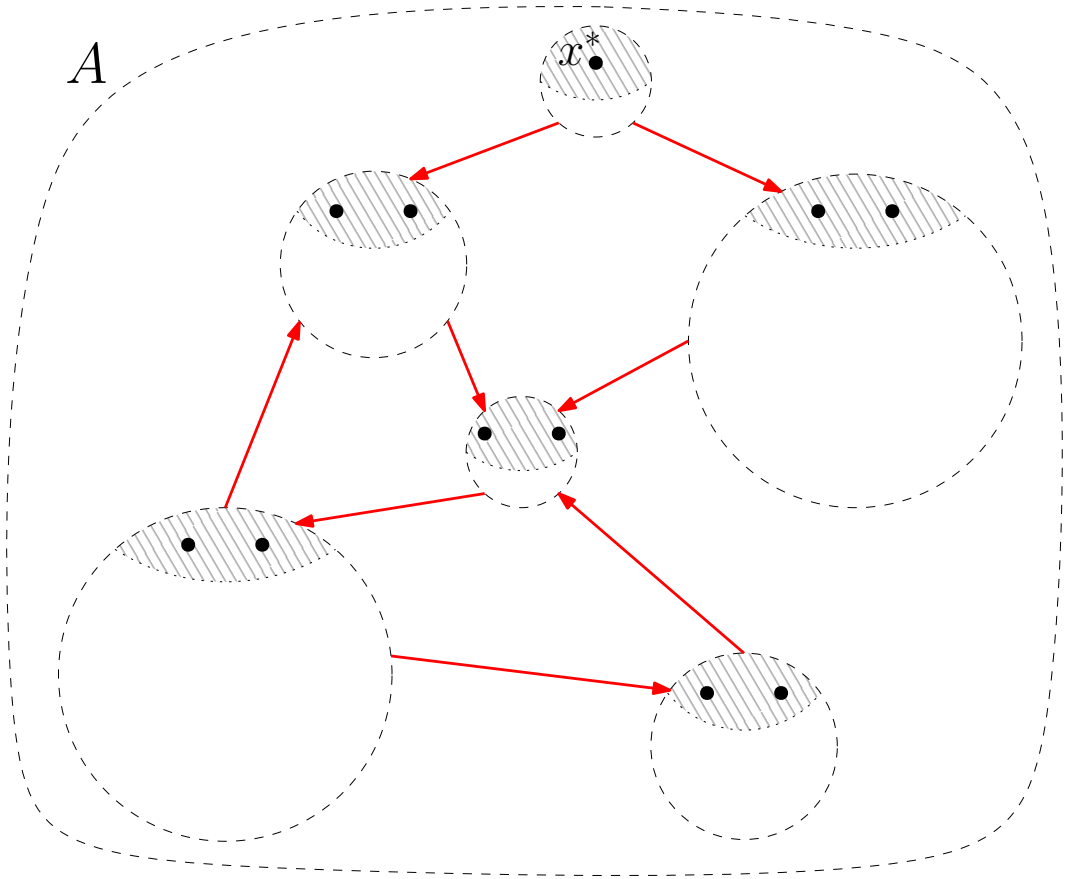
The idea for the algorithm

The inseparable pairs are not enough information on their own. We need information about how they are connected to each other.

The idea for the algorithm

The inseparable pairs are not enough information on their own. We need information about how they are connected to each other.

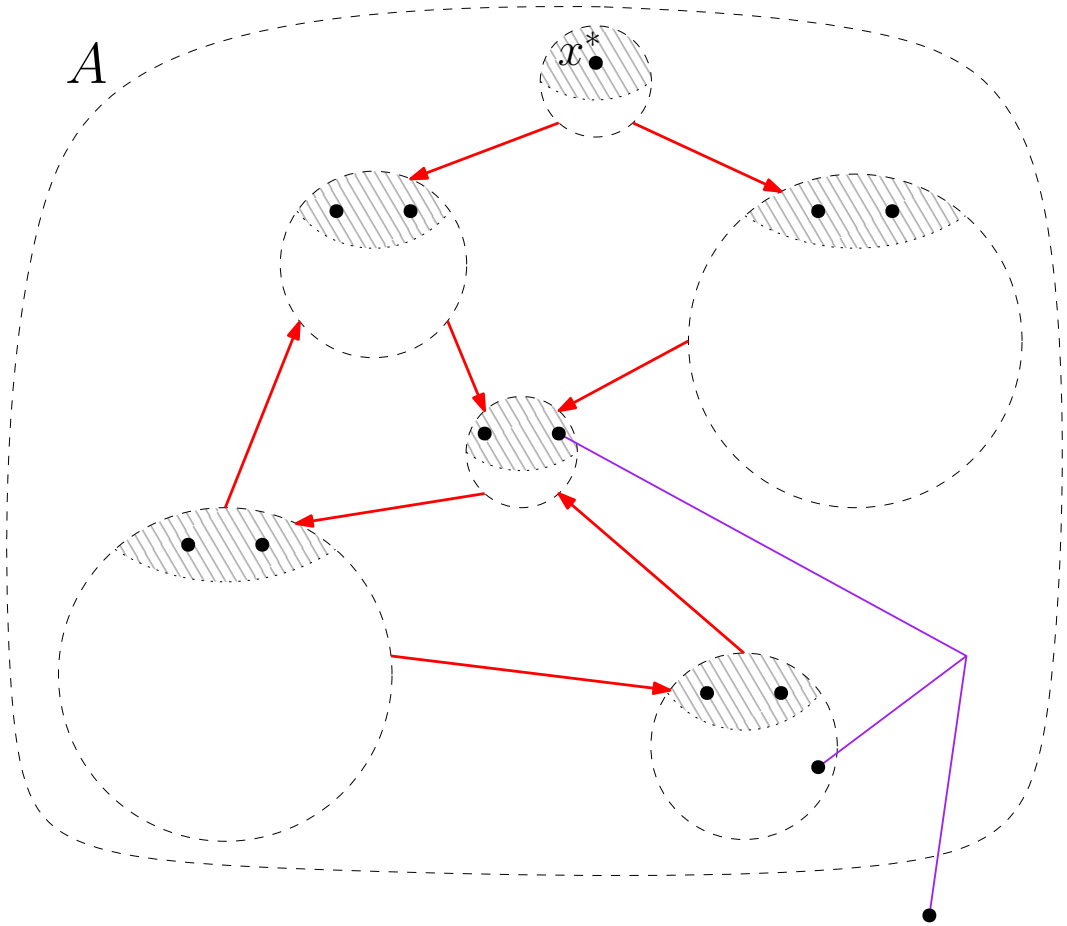
- Solution: the *archipelago* structure.



The idea for the algorithm

The inseparable pairs are not enough information on their own. We need information about how they are connected to each other.

- Solution: the *archipelago* structure.



Main algorithmic result

Let n (resp. m) denote the number of vertices (resp. edges) of H.

- $O(m)$ additions of edges, each in time $O(n)$.
- $O(m^2)$ rejections of edges, each in time $O(1)$.

Main algorithmic result

Let n (resp. m) denote the number of vertices (resp. edges) of H.

- $O(m)$ additions of edges, each in time $O(n)$.
- $O(m^2)$ rejections of edges, each in time $O(1)$.

Theorem [G., Gravier, Sivignon (2022)]

There exists an algorithm which, given a 3-uniform hypergraph and a vertex x^*, computes the linear connected component of x^* in time $O(m^2)$.

Main algorithmic result

Let n (resp. m) denote the number of vertices (resp. edges) of H.

- $O(m)$ additions of edges, each in time $O(n)$.
- $O(m^2)$ rejections of edges, each in time $O(1)$.

Theorem [G., Gravier, Sivignon (2022)]

There exists an algorithm which, given a 3-uniform hypergraph and a vertex x^*, computes the linear connected component of x^* in time $O(m^2)$.

Corollary

MakerBreaker is in polynomial time on hypergraphs of rank 3.

The Maker-Breaker game on hypergraphs of rank 3

1. Introduction
2. Structural result
3. Algorithmic result
4. Conclusion

Summary

- On hypergraphs of rank 3, we have obtained:
  - A structural characterization for the outcome, and a description of both players' optimal strategies, based on intersections of some subhypergraph collections.
  - A polynomial-time algorithm to decide the outcome.

Summary

- On hypergraphs of rank 3, we have obtained:
  - A structural characterization for the outcome, and a description of both players' optimal strategies, based on intersections of some subhypergraph collections.
  - A polynomial-time algorithm to decide the outcome.
- What about hypergraphs of rank 4?

Summary

- On hypergraphs of rank 3, we have obtained:
  - A structural characterization for the outcome, and a description of both players' optimal strategies, based on intersections of some subhypergraph collections.
  - A polynomial-time algorithm to decide the outcome.
- What about hypergraphs of rank 4?
  - Unions of simple structures quickly become very complicated.
  - If trying to prove similar results, a different approach would likely be needed.

Summary

- On hypergraphs of rank 3, we have obtained:
  - A structural characterization for the outcome, and a description of both players' optimal strategies, based on intersections of some subhypergraph collections.
  - A polynomial-time algorithm to decide the outcome.

- What about hypergraphs of rank 4?
  - Unions of simple structures quickly become very complicated.
  - If trying to prove similar results, a different approach would likely be needed.
  - Instead, one could try to prove that:

Conjecture

MakerBreaker is PSPACE-complete on 4-uniform hypergraphs.

Other works: optimization parameters

What if Maker wants to optimize her wins?

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.

Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.

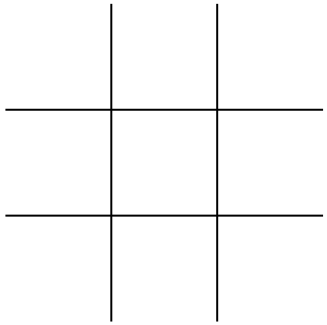
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



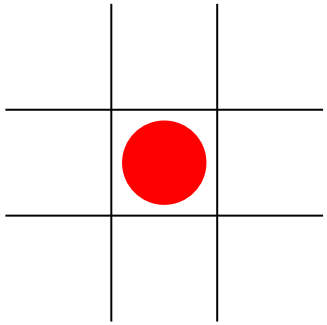
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



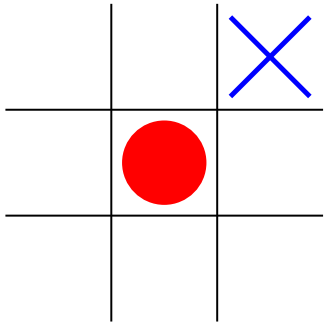
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



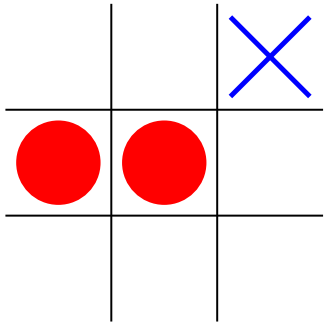
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



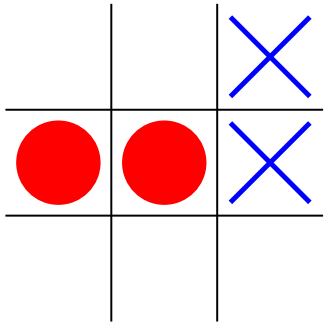
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



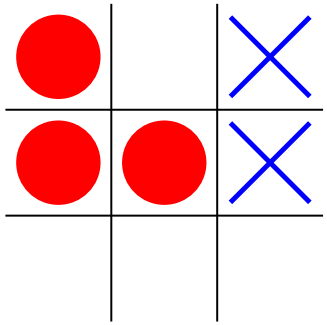
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



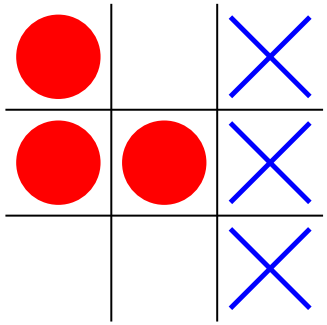
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



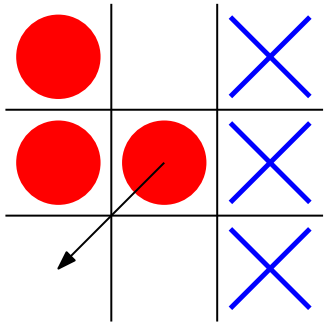
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



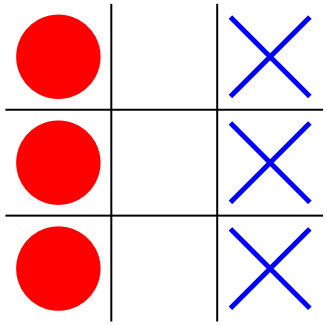
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



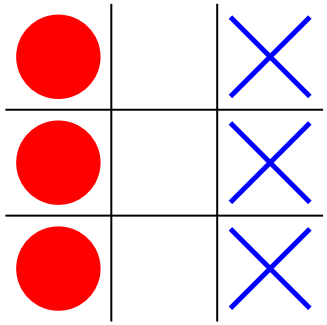
Rank	Number of rounds	
2	2	
3	$\Theta(\log(n))$	
4	$\Theta(n)$	

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



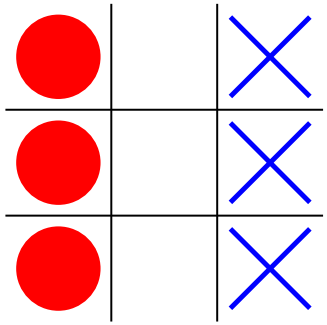
Rank	Number of rounds	Number of tokens
2	2	2
3	$\Theta(\log(n))$	3
4	$\Theta(n)$	$\Theta(n)$

Table: Worst case for Maker wins on n vertices

Other works: optimization parameters

What if Maker wants to optimize her wins?

- Minimizing the number of **rounds**: win as quickly as possible.
- Minimizing the number of **tokens**: in a version of the game where, instead of coloring vertices permanently, Maker places red tokens which she can move around.



Rank	Number of rounds	Number of tokens
2	2	2
3	$\Theta(\log(n))$	3
4	$\Theta(n)$	$\Theta(n)$

Table: Worst case for Maker wins on n vertices

- This tends to confirm the **complexity gap** from rank 3 to rank 4.

Future works?

- **Structural** studies of positional games, especially on hypergraphs with small edges.
- **Polynomial-time** algorithms.

Future works?

- **Structural** studies of positional games, especially on hypergraphs with small edges.
- **Polynomial-time** algorithms.

- Maker-Maker game on hypergraphs of rank 3.

Future works?

- **Structural** studies of positional games, especially on hypergraphs with small edges.
- **Polynomial-time** algorithms.

- Maker-Maker game on hypergraphs of rank 3.
- Avoider-Enforcer game on hypergraphs of rank 3.
  (The edges are losing sets! Avoider wants to avoid getting a monochromatic edge of her color, while Enforcer tries to force her.)

Future works?

- **Structural** studies of positional games, especially on hypergraphs with small edges.
- **Polynomial-time** algorithms.

- Maker-Maker game on hypergraphs of rank 3.
- Avoider-Enforcer game on hypergraphs of rank 3.
  (The edges are losing sets! Avoider wants to avoid getting a monochromatic edge of her color, while Enforcer tries to force her.)
- Unified achievement games: what if both players are "Maker"... but have different winning sets?

Future works?

- **Structural** studies of positional games, especially on hypergraphs with small edges.
- **Polynomial-time** algorithms.

- Maker-Maker game on hypergraphs of rank 3.
- Avoider-Enforcer game on hypergraphs of rank 3.
  (The edges are losing sets! Avoider wants to avoid getting a monochromatic edge of her color, while Enforcer tries to force her.)
- Unified achievement games: what if both players are "Maker"... but have different winning sets?
  Examples: $E_{\text{Bob}} = E_{\text{Alice}} \leftrightarrow$ Maker-Maker.
  $E_{\text{Bob}} =$ transversals of $E_{\text{Alice}} \leftrightarrow$ Maker-Breaker.

Future works?

- **Structural** studies of positional games, especially on hypergraphs with small edges.
- **Polynomial-time** algorithms.

- Maker-Maker game on hypergraphs of rank 3.
- Avoider-Enforcer game on hypergraphs of rank 3.
  (The edges are losing sets! Avoider wants to avoid getting a monochromatic edge of her color, while Enforcer tries to force her.)
- Unified achievement games: what if both players are "Maker"... but have different winning sets?
  Examples: $E_{\text{Bob}} = E_{\text{Alice}} \leftrightarrow$ Maker-Maker.
  $E_{\text{Bob}} =$ transversals of $E_{\text{Alice}} \leftrightarrow$ Maker-Breaker.
  A possible start: all winning sets of size 2?

Merci !