Course 4: Combinatorial Game Theory

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Summary

Last session

1. Unsupervised learning - discover structure from unlabeled data
2. Clustering
3. Decomposition - sparse dictionary learning

Today's session

- Combinatorial Game Theory

Examples

	perfect information	imperfect information
sequential		
concurrent		

Before we proceed...

- Game theory is a very rich and broad scientific domain,
- In economics, we are interested in equilibriums and payoff games,
- In mathematics, we are interested in showing existence of objects,
- And many others...
- We focus in this course on very particular games (sequential with perfect information) under the scope of combinatorial game theory.

  Following the literature, we consider two players called Eve and Adam.

Before we proceed...

- Game theory is a very rich and broad scientific domain,
- In economics, we are interested in equilibriums and payoff games,
- In mathematics, we are interested in showing existence of objects,
- And many others...
- We focus in this course on very particular games (sequential with perfect information) under the scope of combinatorial game theory.

  Following the literature, we consider two players called Eve and Adam.

Before we proceed...

- Game theory is a very rich and broad scientific domain,
- In economics, we are interested in equilibriums and payoff games,
- In mathematics, we are interested in showing existence of objects,
- And many others...
- We focus in this course on very particular games (sequential with perfect information) under the scope of combinatorial game theory.

  Following the literature, we consider two players called Eve and Adam.

Before we proceed...

- Game theory is a very rich and broad scientific domain,
- In economics, we are interested in equilibriums and payoff games,
- In mathematics, we are interested in showing existence of objects,
- And many others...
- We focus in this course on very particular games (sequential with perfect information) under the scope of combinatorial game theory.

  Following the literature, we consider two players called Eve and Adam.

Before we proceed...

- Game theory is a very rich and broad scientific domain,
- In economics, we are interested in equilibriums and payoff games,
- In mathematics, we are interested in showing existence of objects,
- And many others...
- We focus in this course on very particular games (sequential with perfect information) under the scope of combinatorial game theory.

  Following the literature, we consider two players called Eve and Adam.

Before we proceed...

- Game theory is a very rich and broad scientific domain,
- In economics, we are interested in equilibriums and payoff games,
- In mathematics, we are interested in showing existence of objects,
- And many others...
- We focus in this course on very particular games (sequential with perfect information) under the scope of combinatorial game theory.

  Following the literature, we consider two players called Eve and Adam.

Arena

Graph

- A graph G is a pair $\langle V, E \rangle$ where V is the finite set of vertices and $E \subseteq V \times V$ is the set of edges,
- An **arena** is a triple $\langle G, V_E, V_A \rangle$ where V_E, V_A is a bipartition of V,
- We suppose each vertex in V is associated with at least one outgoing edge.
- We will also use the term "state" when referring to vertices of V.

Beware not to confuse an arena with a bipartite graph.

Graph

- A graph G is a pair $\langle V, E\rangle$ where V is the finite set of vertices and $E \subseteq V \times V$ is the set of edges,
- An **arena** is a triple $\langle G, V_E, V_A\rangle$ where V_E, V_A is a bipartition of V,
- We suppose each vertex in V is associated with at least one outgoing edge.
- We will also use the term "state" when referring to vertices of V.

Beware not to confuse an arena with a bipartite graph.

Arena

Graph

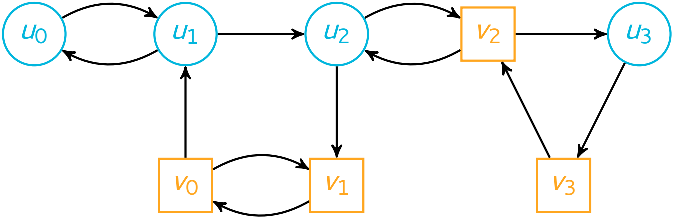
- A graph G is a pair $\langle V, E \rangle$ where V is the finite set of vertices and $E \subseteq V \times V$ is the set of edges,
- An **arena** is a triple $\langle G, V_E, V_A \rangle$ where V_E, V_A is a bipartition of V,
- We suppose each vertex in V is associated with at least one outgoing edge.
- We will also use the term "state" when referring to vertices of V.

Beware not to confuse an arena with a bipartite graph.

Arena

Graph

- A graph G is a pair $\langle V, E \rangle$ where V is the finite set of vertices and $E \subseteq V \times V$ is the set of edges,
- An **arena** is a triple $\langle G, V_E, V_A \rangle$ where V_E, V_A is a bipartition of V,
- We suppose each vertex in V is associated with at least one outgoing edge.
- We will also use the term "state" when referring to vertices of V.

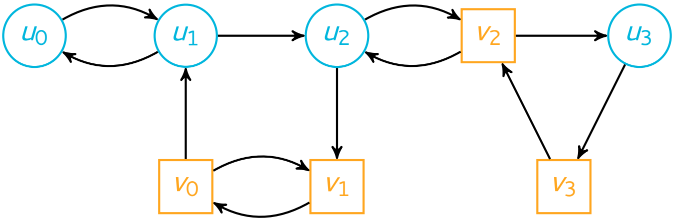


Beware not to confuse an arena with a bipartite graph.

Arena

Graph

- A graph G is a pair $\langle V, E \rangle$ where V is the finite set of vertices and $E \subseteq V \times V$ is the set of edges,
- An **arena** is a triple $\langle G, V_E, V_A \rangle$ where V_E, V_A is a bipartition of V,
- We suppose each vertex in V is associated with at least one outgoing edge.
- We will also use the term "state" when referring to vertices of V.

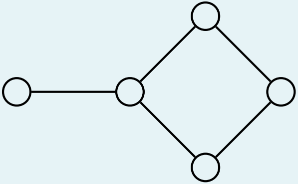


Beware not to confuse an arena with a bipartite graph.

Example game

Cops and robbers

- Consider the following graph:

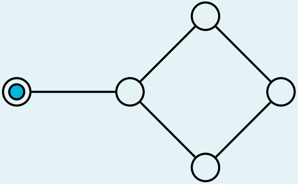


- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

Example game

Cops and robbers

- Consider the following graph:

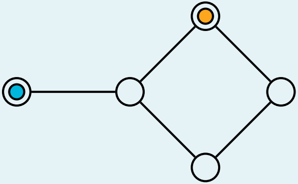


- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

Example game

Cops and robbers

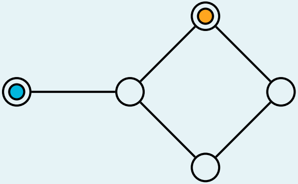
- Consider the following graph:



- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

Cops and robbers

- Consider the following graph:

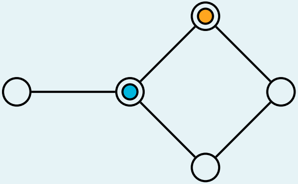


- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

Example game

Cops and robbers

- Consider the following graph:

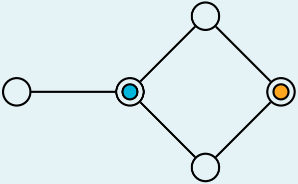


- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

Example game

Cops and robbers

- Consider the following graph:

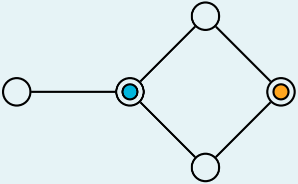


- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

Example game

Cops and robbers

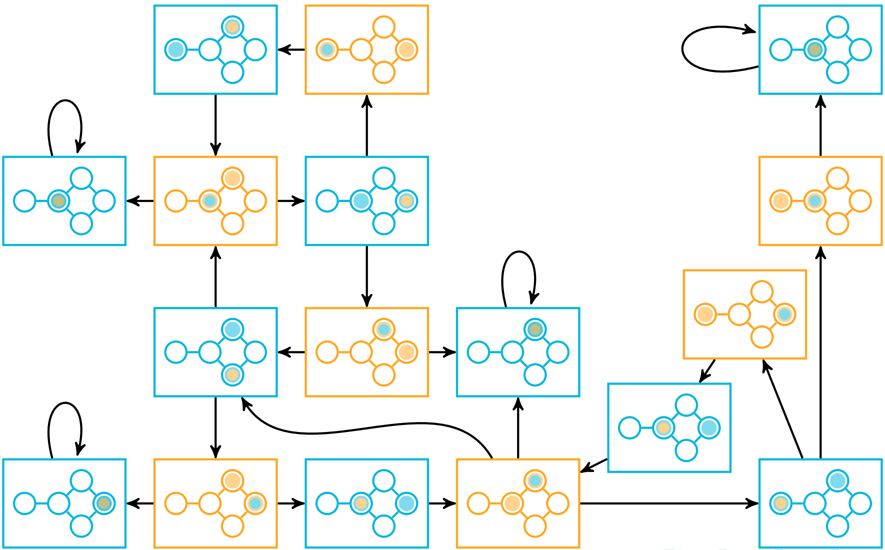
- Consider the following graph:



- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

Example game

Partial arena (symmetric configurations are not represented)



Playout and winning condition

Playout

- A **playout** λ is an infinite walk on G,
- The initial vertex of a playout is called the **starting position**.

Winning conditions

Denote $F \subseteq V$ a set of final states. Eve wins a playout *if and only if*:

- **Reachability:** λ goes through at least one final state (e.g. Go),
- **Co-Reachability:** λ never goes through a final state (e.g. model-checking),

Other reachability conditions exist (Büchi, co-Büchi) , in which we consider final states finitely / infinitely often.
Note that in most practical cases in AI, reachability is considered.

Playout and winning condition

Playout

- A **playout** λ is an infinite walk on G,
- The initial vertex of a playout is called the **starting position**.

Winning conditions

Denote $F \subseteq V$ a set of final states. Eve wins a playout *if and only if*:

- **Reachability:** λ goes through at least one final state (e.g. Go),
- **Co-Reachability:** λ never goes through a final state (e.g. model-checking),

Other reachability conditions exist (Büchi, co-Büchi) , in which we consider final states finitely / infinitely often.
Note that in most practical cases in AI, reachability is considered.

Playout and winning condition

Playout

- A **playout** λ is an infinite walk on G,
- The initial vertex of a playout is called the **starting position**.

Winning conditions

Denote $F \subseteq V$ a set of final states. Eve wins a playout *if and only if*:

- **Reachability:** λ goes through at least one final state (e.g. Go),
- **Co-Reachability:** λ never goes through a final state (e.g. model-checking),

Other reachability conditions exist (Büchi, co-Büchi) , in which we consider final states finitely / infinitely often.
Note that in most practical cases in AI, reachability is considered.

Playout and winning condition

Playout

- A **playout** λ is an infinite walk on G,
- The initial vertex of a playout is called the **starting position**.

Winning conditions

Denote $F \subseteq V$ a set of final states. Eve wins a playout *if and only if*:

- **Reachability:** λ goes through at least one final state (e.g. Go),
- **Co-Reachability:** λ never goes through a final state (e.g. model-checking),

Other reachability conditions exist (Büchi, co-Büchi) , in which we consider final states finitely / infinitely often.
Note that in most practical cases in AI, reachability is considered.

Playout and winning condition

Playout

- A **playout** λ is an infinite walk on G,
- The initial vertex of a playout is called the **starting position**.

Winning conditions

Denote $F \subseteq V$ a set of final states. Eve wins a playout *if and only if*:

- **Reachability:** λ goes through at least one final state (e.g. Go),
- **Co-Reachability:** λ never goes through a final state (e.g. model-checking),

Other reachability conditions exist (Büchi, co-Büchi) , in which we consider final states finitely / infinitely often.

Note that in most practical cases in AI, reachability is considered.

Playout and winning condition

Playout

- A **playout** λ is an infinite walk on G,
- The initial vertex of a playout is called the **starting position**.

Winning conditions

Denote $F \subseteq V$ a set of final states. Eve wins a playout *if and only if*:

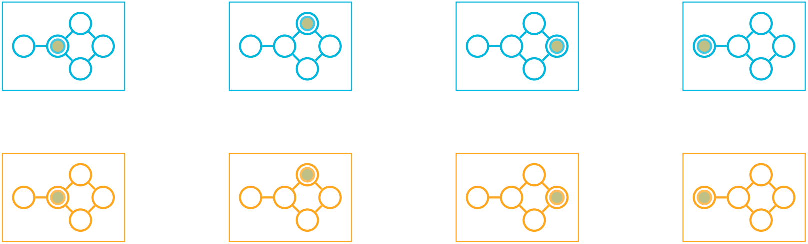
- **Reachability:** λ goes through at least one final state (e.g. Go),
- **Co-Reachability:** λ never goes through a final state (e.g. model-checking),

Other reachability conditions exist (Büchi, co-Büchi) , in which we consider final states finitely / infinitely often.
Note that in most practical cases in AI, reachability is considered.

The case of cops and robbers

Here the winning condition is of type reachability for the cop and co-reachability for the robber.

Final states are:



Strategy

Strategy

- A **strategy** for Eve is a partial function $\phi_E : V^* \to V$,
- A **randomized strategy** is a partial function $\phi_E : V^* \to P(V)$, where $P(V)$ is the set of probability distributions over V.

Induced playout

- An **induced playout** associated with ϕ_E and ϕ_A is a playout λ such that:

$$\forall i > 0, \lambda_i = \begin{cases} \phi_E(\lambda_0 \ldots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_E \\ \phi_A(\lambda_0 \ldots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_A \end{cases},$$

  we write it $\lambda(\phi_E, \phi_A, V_0)$, where V_0 is the starting position.
- For randomized strategies, one can make use of the *Carathéodory's extension theorem*.

Strategy

Strategy

- A **strategy** for Eve is a partial function $\phi_E : V^* \to V$,
- A **randomized strategy** is a partial function $\phi_E : V^* \to P(V)$, where $P(V)$ is the set of probability distributions over V.

Induced playout

- An **induced playout** associated with ϕ_E and ϕ_A is a playout λ such that:

$$\forall i > 0, \lambda_i = \begin{cases} \phi_E(\lambda_0 \dots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_E \\ \phi_A(\lambda_0 \dots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_A \end{cases},$$

  we write it $\lambda(\phi_E, \phi_A, V_0)$, where V_0 is the starting position.
- For randomized strategies, one can make use of the *Carathéodory's extension theorem*.

Strategy

Strategy

- A **strategy** for Eve is a partial function $\phi_E : V^* \to V$,
- A **randomized strategy** is a partial function $\phi_E : V^* \to P(V)$, where $P(V)$ is the set of probability distributions over V.

Induced playout

- An **induced playout** associated with ϕ_E and ϕ_A is a playout λ such that:

$$\forall i > 0, \lambda_i = \begin{cases} \phi_E(\lambda_0 \ldots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_E \\ \phi_A(\lambda_0 \ldots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_A \end{cases},$$

we write it $\lambda(\phi_E, \phi_A, V_0)$, where V_0 is the starting position.

- For randomized strategies, one can make use of the *Carathéodory's extension theorem*.

Strategy

Strategy

- A **strategy** for Eve is a partial function $\phi_E : V^* \to V$,
- A **randomized strategy** is a partial function $\phi_E : V^* \to P(V)$, where $P(V)$ is the set of probability distributions over V.

Induced playout

- An **induced playout** associated with ϕ_E and ϕ_A is a playout λ such that:

$$\forall i > 0, \lambda_i = \begin{cases} \phi_E(\lambda_0 \ldots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_E \\ \phi_A(\lambda_0 \ldots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_A \end{cases},$$

we write it $\lambda(\phi_E, \phi_A, V_0)$, where V_0 is the starting position.
- For randomized strategies, one can make use of the *Carathéodory's extension theorem*.

Winning strategies and memory

Winning strategy

- A strategy ϕ_E for Eve is said to be winning from $V_0 \in V$ if:

$$\forall \phi_A, \lambda(\phi_E, \phi_A, V_0) \text{ is winning for Eve}$$

- For randomized strategies, we are typically interested in almost-surely winning strategies.

Positional strategy

- A strategy ϕ_E for Eve is said positional if $\exists \phi_E^P : V \to V$ such that

$$\forall i \in \mathbb{N}, \forall V_0 V_1 \dots V_{i-1} \in V^i, \forall V_i \in V_E,$$
$$\phi_E(V_0 V_1 \dots V_{i-1} V_i) = \phi_E^P(V_i).$$

A positional strategy is sometimes termed "without memory".

Winning strategies and memory

Winning strategy

- A strategy ϕ_E for Eve is said to be winning from $V_0 \in V$ if:

$$\forall \phi_A, \lambda(\phi_E, \phi_A, V_0) \text{ is winning for Eve}$$

- For randomized strategies, we are typically interested in almost-surely winning strategies.

Positional strategy

- A strategy ϕ_E for Eve is said positional if $\exists \phi_E^p : V \to V$ such that

$$\forall i \in \mathbb{N}, \forall V_0 V_1 \dots V_{i-1} \in V^i, \forall V_i \in V_E,$$
$$\phi_E(V_0 V_1 \dots V_{i-1} V_i) = \phi_E^p(V_i).$$

A positional strategy is sometimes termed "without memory".

Winning strategies and memory

Winning strategy

- A strategy ϕ_E for Eve is said to be winning from $V_0 \in V$ if:

$$\forall \phi_A, \lambda(\phi_E, \phi_A, V_0) \text{ is winning for Eve}$$

- For randomized strategies, we are typically interested in almost-surely winning strategies.

Positional strategy

- A strategy ϕ_E for Eve is said positional if $\exists \phi_E^p : V \to V$ such that

$$\forall i \in \mathbb{N}, \forall V_0 V_1 \ldots V_{i-1} \in V^i, \forall V_i \in V_E,$$
$$\phi_E(V_0 V_1 \ldots V_{i-1} V_i) = \phi_E^p(V_i).$$

A positional strategy is sometimes termed "without memory".

Determined games

Game

- A **game** $\mathbb{G}$ is a tuple $\langle G, V_E, V_A, F, W\rangle$, where
  - $\langle G = \langle V, E\rangle, V_E, V_A\rangle$ is an arena,
  - $F \subseteq V$ is the set of final states,
  - W is a winning condition.

Determined games

- A game is said **determined** if for each starting position, either Eve or Adam admits a winning strategy.

Theorem 1

- All games considered in this course are determined,
- Moreover, winning strategies can always be chosen positional.

Game

- A **game** $\mathbb{G}$ is a tuple $\langle G, V_E, V_A, F, W\rangle$, where
  - $\langle G = \langle V, E\rangle, V_E, V_A\rangle$ is an arena,
  - $F \subseteq V$ is the set of final states,
  - W is a winning condition.

Determined games

- A game is said **determined** if for each starting position, either Eve or Adam admits a winning strategy.

Theorem 1

- All games considered in this course are determined,
- Moreover, winning strategies can always be chosen positional.

Determined games

Game

- A **game** $\mathbb{G}$ is a tuple $\langle G, V_E, V_A, F, W\rangle$, where
  - $\langle G = \langle V, E\rangle, V_E, V_A\rangle$ is an arena,
  - $F \subseteq V$ is the set of final states,
  - W is a winning condition.

Determined games

- A game is said **determined** if for each starting position, either Eve or Adam admits a winning strategy.

Theorem 1

- All games considered in this course are determined,
- Moreover, winning strategies can always be chosen positional.

Determined games

Game

- A **game** $\mathbb{G}$ is a tuple $\langle G, V_E, V_A, F, W\rangle$, where
  - $\langle G = \langle V, E\rangle, V_E, V_A\rangle$ is an arena,
  - $F \subseteq V$ is the set of final states,
  - W is a winning condition.

Determined games

- A game is said **determined** if for each starting position, either Eve or Adam admits a winning strategy.

Theorem 1

- All games considered in this course are determined,
- Moreover, winning strategies can always be chosen positional.

Sketch of the proof for reachability games

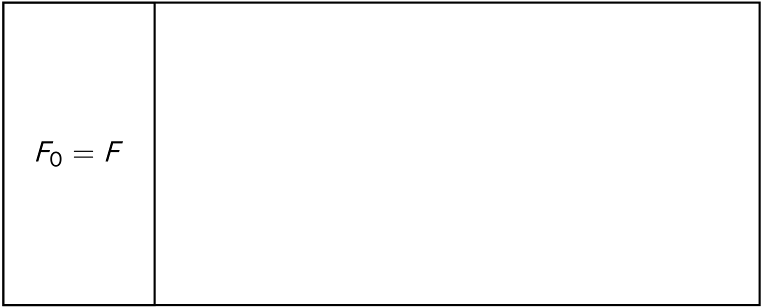
A: Attractor
T: Trap

Arena

Sketch of the proof for reachability games

A: Attractor
T: Trap

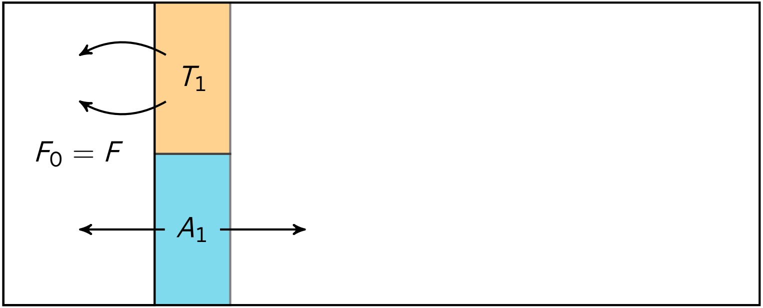
Arena



Sketch of the proof for reachability games

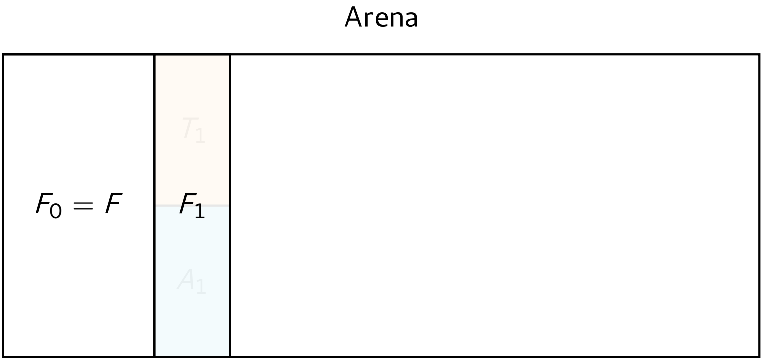
A: Attractor
T: Trap

Arena



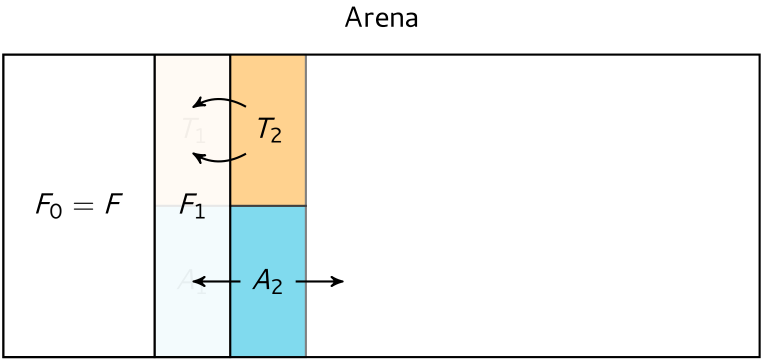
Sketch of the proof for reachability games

A: Attractor
T: Trap



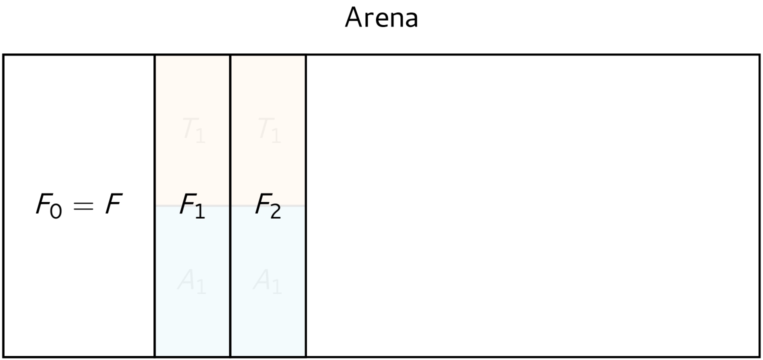
Sketch of the proof for reachability games

A: Attractor
T: Trap



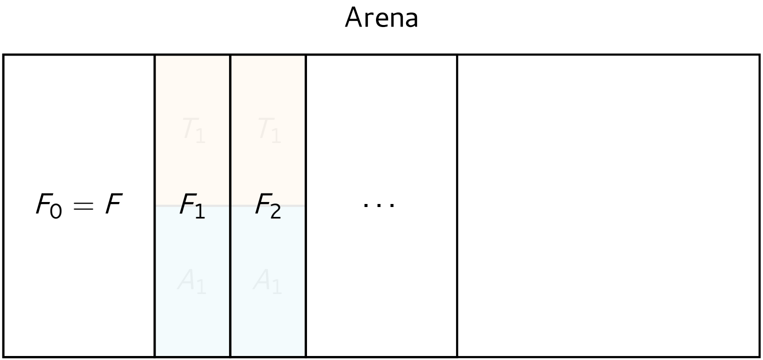
Sketch of the proof for reachability games

A : Attractor
T : Trap



Sketch of the proof for reachability games

A: Attractor
T: Trap



Sketch of the proof for reachability games

A : Attractor
T : Trap

Arena

$$\bigcup_{i=0}^{+\infty} F_i$$

$$V - \bigcup_{i=0}^{+\infty} F_i$$

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena

$\bigcup_{i=0}^{+\infty} F_i$ Winning region for Eve	$V - \bigcup_{i=0}^{+\infty} F_i$ Winning region for Adam

Illustration on the example

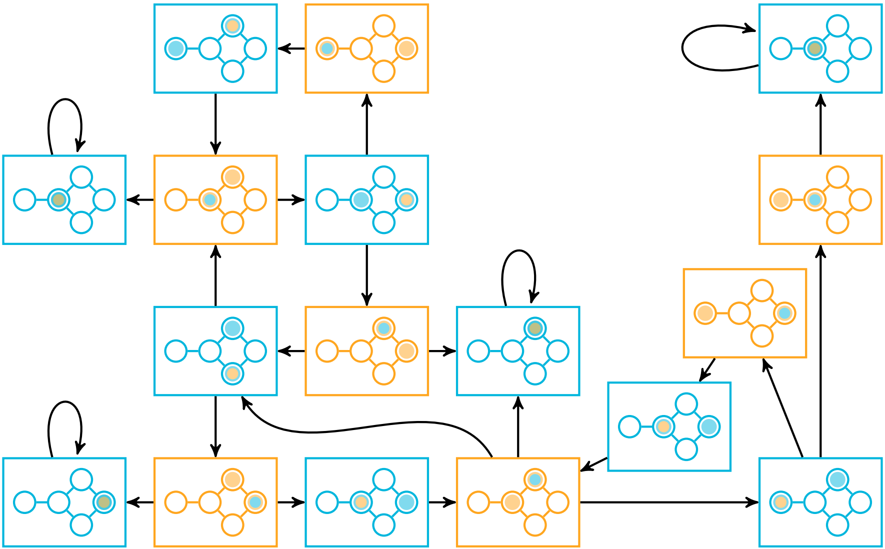


Illustration on the example

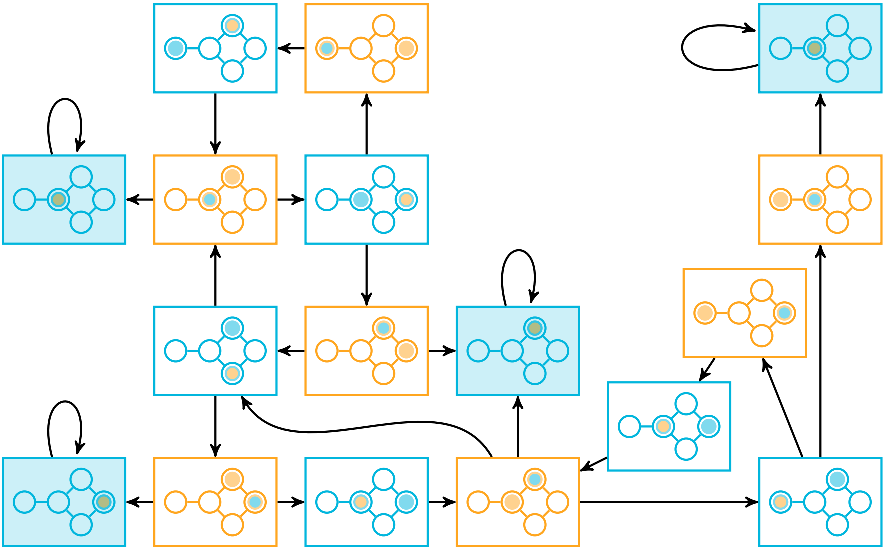


Illustration on the example

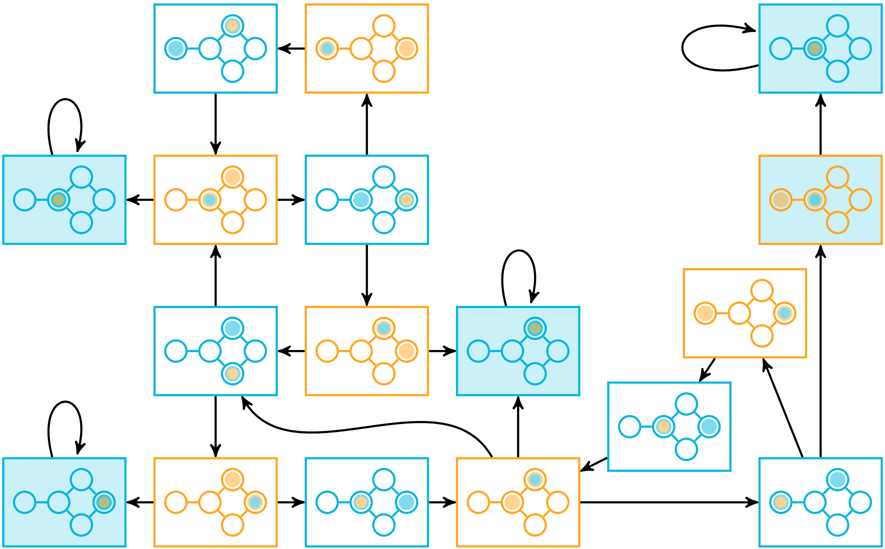


Illustration on the example

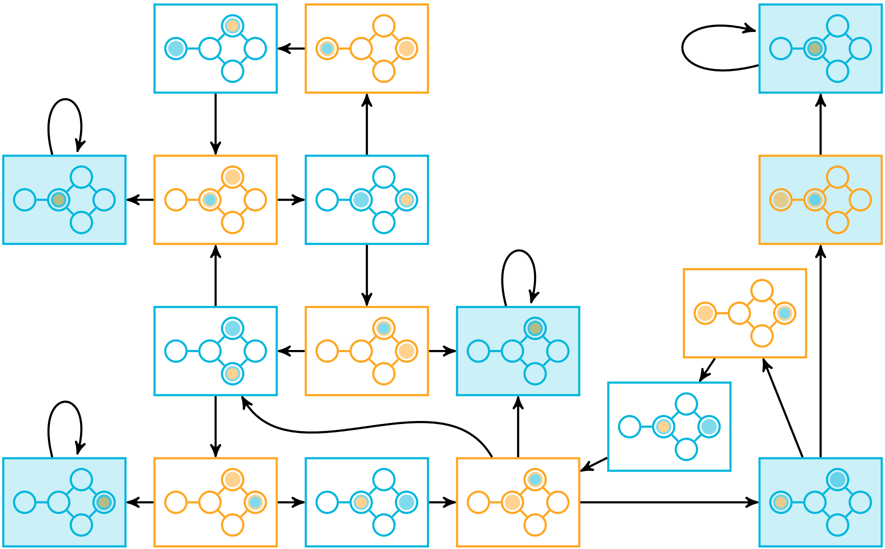
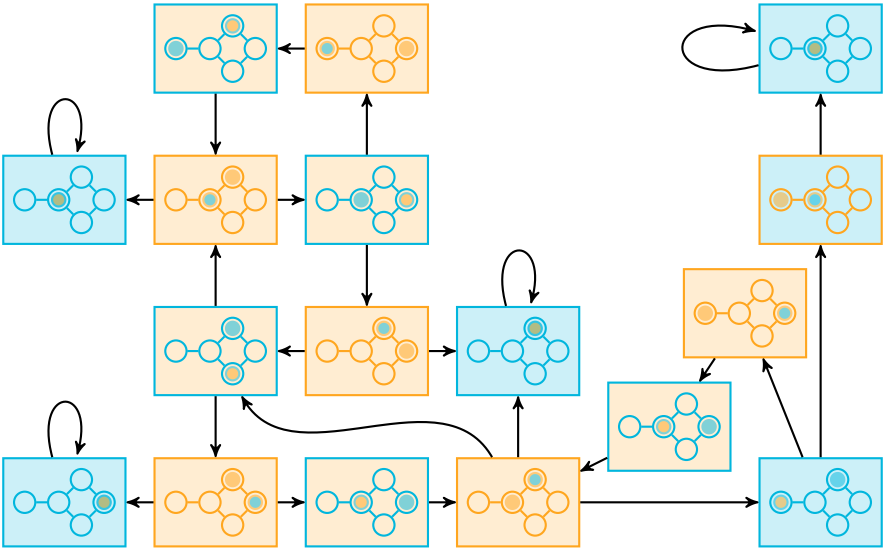


Illustration on the example

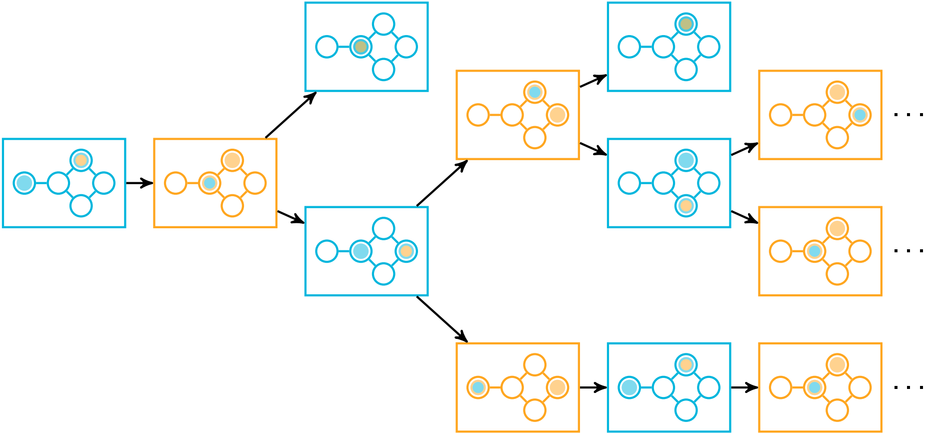


Playout tree

- The **playout tree** rooted at vertex V_0 is the infinite tree of all possible plays starting at V_0.

Playout tree

- The **playout tree** rooted at vertex V_0 is the infinite tree of all possible plays starting at V_0.



Playout tree

Playout tree

- The **playout tree** rooted at vertex V_0 is the infinite tree of all possible plays starting at V_0.

Finding winning strategies

- In practice, it is possible to use the construction of the proof of Theorem 1 to find winning strategies,
- When V is too large, it may be better to search the playout tree.

Exploring large playout trees

- Even when playouts are finite, playouts trees can quickly become untractably large,
- Randomly explore to find interesting strategies,
- A possible such method is Monte-Carlo Tree-Search (MCTS) or to derive machine learning strategies.

Playout tree

Playout tree

- The **playout tree** rooted at vertex V_0 is the infinite tree of all possible plays starting at V_0.

Finding winning strategies

- In practice, it is possible to use the construction of the proof of Theorem 1 to find winning strategies,
- When V is too large, it may be better to search the playout tree.

Exploring large playout trees

- Even when playouts are finite, playouts trees can quickly become untractably large,
- Randomly explore to find interesting strategies,
- A possible such method is Monte-Carlo Tree-Search (MCTS) or to derive machine learning strategies.

Lab Session 4

TP Combinatorial Game Theory (TP3)

- Pyrat game with the python playing using a greedy approach (closest cheese)
- Program an exhaustive playout tree search for the rat to beat the python

Challenge annoucement!

Solo pyrat game against a Greedy algorithm, followed by a tournament on December 5th.

- Complete rules and modalities (deadline, etc) are on Moodle
- Baseline with supervised learning
- Oral presentation (10+5 minutes) of your solution

More details on Moodle (section : Challenge Information).