Course 4: Combinatorial Game Theory

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

2018-11-19

Course 4: Combinatorial Game Theory

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

Summary
Last session
1 Unsupervised learning - discover structure from unlabeled data
2 Clustering
3 Decomposition - sparse dictionary learning
Today's session
Combinatorial Game Theory

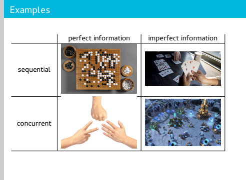
Examples

	perfect information	imperfect information
sequential		
concurrent		

2018-11-19

Course 4: Combinatorial Game Theory

└Examples



We can derive two by two different types of games. Sequential vs concurrent, and Perfect vs imperfect information. In a sequential game, players play turn by turn. In a concurrent game, players play simulataneously.

In a game with perfect information, all players have access to the same information at the same time, all the information is visible. However, in a game with imperfect information, there is some hidden data / information, for example the hidden cards in a game of poker, or the invisible parts of the map in a StarCraft game.

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.

The goal of this slide is to explain that there are other possible formulations of "game theory", so we are focusing here on a specific part. We can mention here that in the next lesson on reinforcment learning, we will also consider the notion of reward in games.

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.

The goal of this slide is to explain that there are other possible formulations of "game theory", so we are focusing here on a specific part. We can mention here that in the next lesson on reinforcment learning, we will also consider the notion of reward in games.

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.

The goal of this slide is to explain that there are other possible formulations of "game theory", so we are focusing here on a specific part. We can mention here that in the next lesson on reinforcment learning, we will also consider the notion of reward in games.

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.

The goal of this slide is to explain that there are other possible formulations of "game theory", so we are focusing here on a specific part. We can mention here that in the next lesson on reinforcment learning, we will also consider the notion of reward in games.

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.

The goal of this slide is to explain that there are other possible formulations of "game theory", so we are focusing here on a specific part. We can mention here that in the next lesson on reinforcment learning, we will also consider the notion of reward in games.

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…

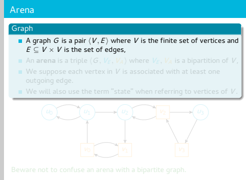
- Game theory is a very rich and broad scientific domain,
- In economics, we are interested in equilibriums and payoff games,
- In mathematics, we are interested in showing existence of objects,
- And many others…
- We focus in this course on very particular games (sequential with perfect information)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

The goal of this slide is to explain that there are other possible formulations of "game theory", so we are focusing here on a specific part. We can mention here that in the next lesson on reinforcment learning, we will also consider the notion of reward in games.

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.

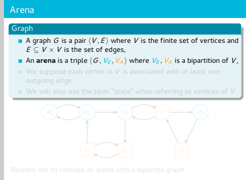


In the figure, the u_i are in V_E, the v_j are in V_A

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.



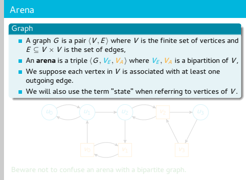
In the figure, the u_i are in V_E, the v_j are in V_A

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.

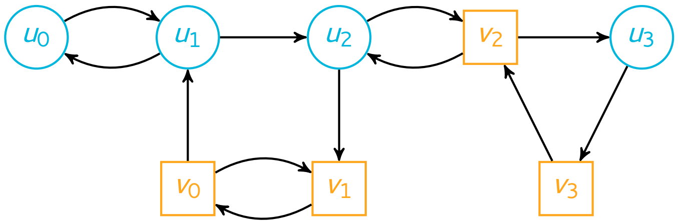


In the figure, the u_i are in V_E, the v_j are in V_A

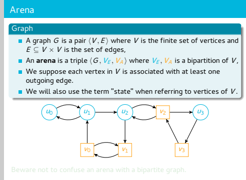
Arena

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.

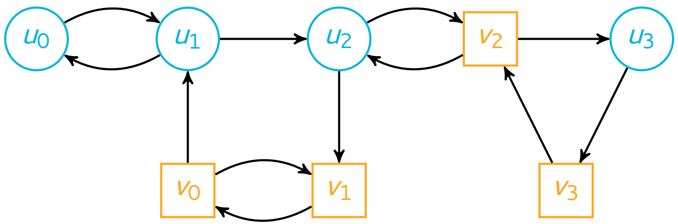


In the figure, the u_i are in V_E, the v_j are in V_A

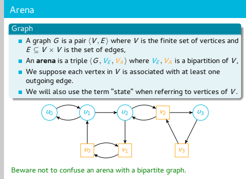
Arena

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.



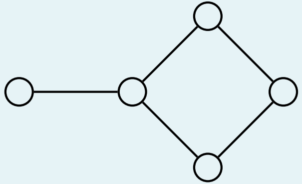
Beware not to confuse an arena with a bipartite graph.



In the figure, the u_i are in V_E, the v_j are in V_A

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.

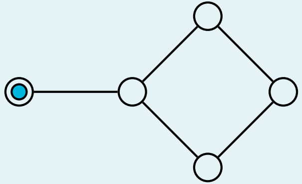
We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

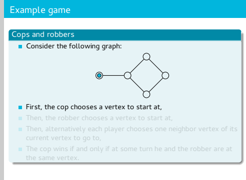
Example game

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.



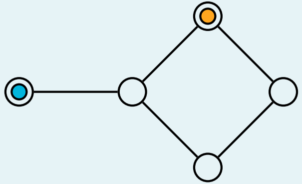
We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

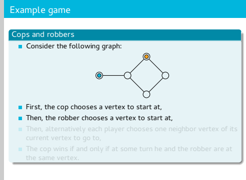
Example game

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.

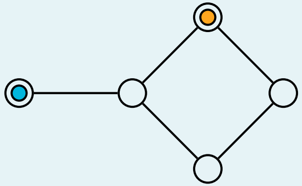


We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

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.

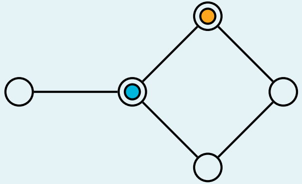
We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

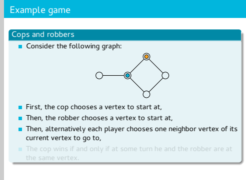
Example game

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.



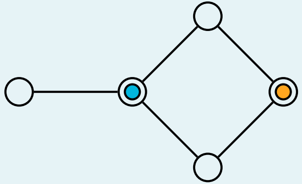
We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

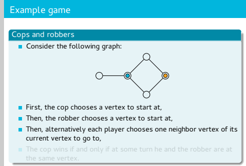
Example game

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.



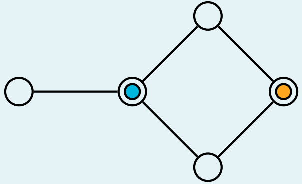
We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

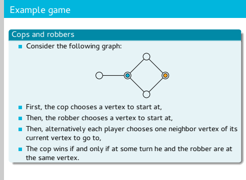
Example game

Cops and robbers

- Consider the following graph:



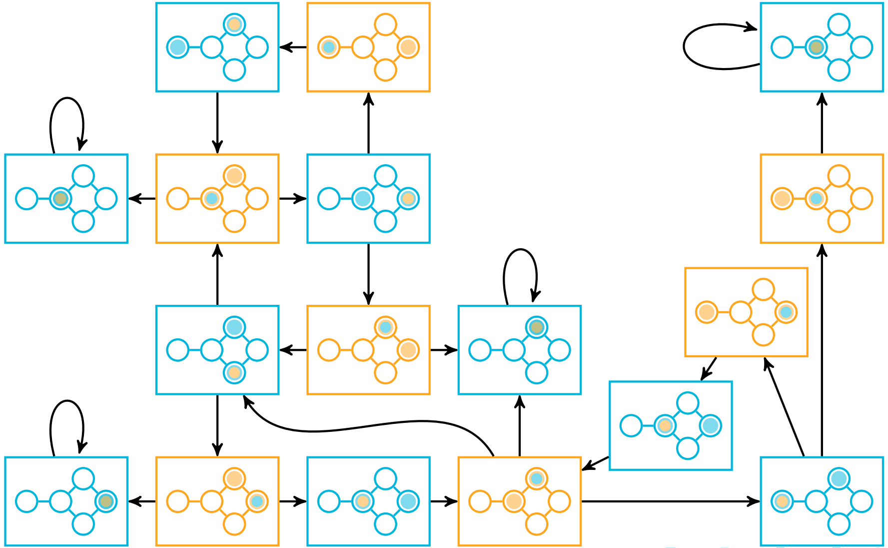
- First, the cop chooses a vertex to start at,
- Then, the robber chooses a vertex to start at,
- Then, alternatively each player chooses one neighbor vertex of its current vertex to go to,
- The cop wins if and only if at some turn he and the robber are at the same vertex.

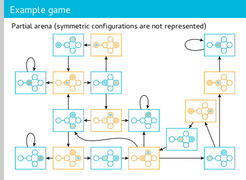


We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

Partial arena (symmetric configurations are not represented)





We give here an example game: the cop and the robber. When the partial arena appears, spend some time explaining starting from the top left node of the arena. The Cop (Eve-blue) has to play and can only go right. We go to the orange node just below. The robber (Adam, orange) has to play, and can choose between going on the same place as the Cop, or it can go on the rightmost place (cell on the right).

Explain here as well that we represent the nodes in whih the Cop has won by putting a arrow pointing towards the same cell.

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 ⊆ 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.

Recall here the definition of a walk on a graph : a walk is a sequence of vertices. So, the same vertex can appear several times in a walk. As said before, when referring to states we refer to vertices of the arena.
About the winning conditions: In the next slide we will explain what it corresponds to for the cops and robber case. (Reachability for the Cop, co-reachability for the Robber).
For those who are interested in knowing more about Büchi and Co-büchi, give the example for Büchi (going through final states infinitely often); here is a situation in which a player has to come back to a certain state periodically, for example to recharge its battery.

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.

Recall here the definition of a walk on a graph : a walk is a sequence of vertices. So, the same vertex can appear several times in a walk.
As said before, when referring to states we refer to vertices of the arena.
About the winning conditions: In the next slide we will explain what it corresponds to for the cops and robber case. (Reachability for the Cop, co-reachability for the Robber).
For those who are interested in knowing more about Büchi and Co-büchi, give the example for Büchi (going through final states infinitely often); here is a situation in which a player has to come back to a certain state periodically, for example to recharge its battery.

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 ⊆ 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.

Recall here the definition of a walk on a graph : a walk is a sequence of vertices. So, the same vertex can appear several times in a walk. As said before, when referring to states we refer to vertices of the arena.
About the winning conditions: In the next slide we will explain what it corresponds to for the cops and robber case. (Reachability for the Cop, co-reachability for the Robber).
For those who are interested in knowing more about Büchi and Co-büchi, give the example for Büchi (going through final states infinitely often); here is a situation in which a player has to come back to a certain state periodically, for example to recharge its battery.

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 ⊆ 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.

Recall here the definition of a walk on a graph : a walk is a sequence of vertices. So, the same vertex can appear several times in a walk.
As said before, when referring to states we refer to vertices of the arena.
About the winning conditions: In the next slide we will explain what it corresponds to for the cops and robber case. (Reachability for the Cop, co-reachability for the Robber).
For those who are interested in knowing more about Büchi and Co-büchi, give the example for Büchi (going through final states infinitely often); here is a situation in which a player has to come back to a certain state periodically, for example to recharge its battery.

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 ⊆ 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.

Recall here the definition of a walk on a graph : a walk is a sequence of vertices. So, the same vertex can appear several times in a walk. As said before, when referring to states we refer to vertices of the arena.
About the winning conditions: In the next slide we will explain what it corresponds to for the cops and robber case. (Reachability for the Cop, co-reachability for the Robber).
For those who are interested in knowing more about Büchi and Co-büchi, give the example for Büchi (going through final states infinitely often); here is a situation in which a player has to come back to a certain state periodically, for example to recharge its battery.

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.

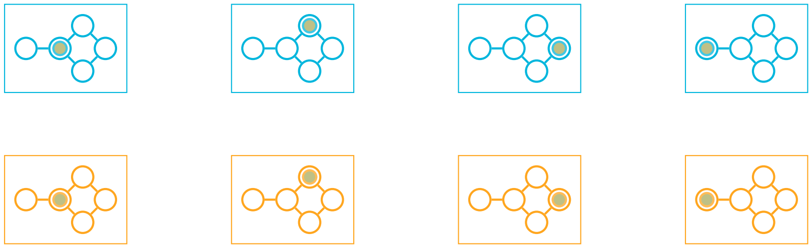
Playout and winning condition
Playout
A playout λ is an infinite walk on G,
The initial vertex of a playout is called the starting position.
Winning conditions
Denote $F \subseteq V$ a set of final states. Eve wins a playout if and only if:
Reachability: λ goes through at least one final state (e.g. 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.

Recall here the definition of a walk on a graph : a walk is a sequence of vertices. So, the same vertex can appear several times in a walk. As said before, when referring to states we refer to vertices of the arena.
About the winning conditions: In the next slide we will explain what it corresponds to for the cops and robber case. (Reachability for the Cop, co-reachability for the Robber).
For those who are interested in knowing more about Büchi and Co-büchi, give the example for Büchi (going through final states infinitely often); here is a situation in which a player has to come back to a certain state periodically, for example to recharge its battery.

The case of cops and robbers

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

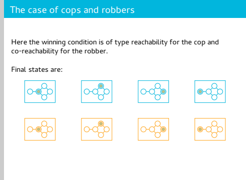
Final states are:



2018-11-19

Course 4: Combinatorial Game Theory

└─The case of cops and robbers



Reachability for the cop, because a winning playout of the cop goes through a final state (the cop catches the robber).
Co-reachability for the robbor, because a winning playout of the robber never goes through a final state (the cop never catches the robber).

Strategy

Strategy

- A **strategy** for Eve is a partial function $\phi_E : V^* \rightarrow V$,
- A **randomized strategy** is a partial function $\phi_E : V^* \rightarrow P(V)$, 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*.

Here, we just want to formalize the concept of strategy and induced playout. A strategy is how to choose the next state based on previous states. An induced playout is what happens when both players follow their 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*.

Here, we just want to formalize the concept of strategy and induced playout. A strategy is how to choose the next state based on previous states. An induced playout is what happens when both players follow their 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:
$\forall i > 0, \lambda_i = \begin{cases} \phi_E(\lambda_0 \dots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_E \\ \phi_A(\lambda_0 \dots \lambda_{i-1}) & \text{if } \lambda_{i-1} \in V_A \end{cases},$
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.

Here, we just want to formalize the concept of strategy and induced playout. A strategy is how to choose the next state based on previous states. An induced playout is what happens when both players follow their strategy.

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

Here, we just want to formalize the concept of strategy and induced playout. A strategy is how to choose the next state based on previous states. An induced playout is what happens when both players follow their strategy.

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”.

A strategy is winning for a player if it can beat all possible strategies of the opponent. A positional strategy is the case in which the current state of the game (arena) is enough to determine the next state, irrespectively of the history of previous states. That is why it is 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:
$\forall \phi_A, \lambda(\phi_E, \phi_A, V_0)$ 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".

A strategy is winning for a player if it can beat all possible strategies of the opponent. A positional strategy is the case in which the current state of the game (arena) is enough to determine the next state, irrespectively of the history of previous states. That is why it is 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”.

A strategy is winning for a player if it can beat all possible strategies of the opponent. A positional strategy is the case in which the current state of the game (arena) is enough to determine the next state, irrespectively of the history of previous states. That is why it is 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.

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.

Now we are ready to formalize games in the framework of combinatorial game theory.

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.

Now we are ready to formalize games in the framework of combinatorial game theory.

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.

Now we are ready to formalize games in the framework of combinatorial game theory.

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.

Now we are ready to formalize games in the framework of combinatorial game theory.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena

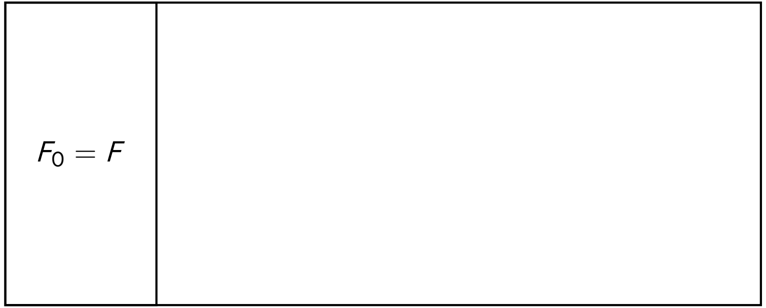
Sketch of the proof for reachability games
A: Attractor
T: Trap
Arena

The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena



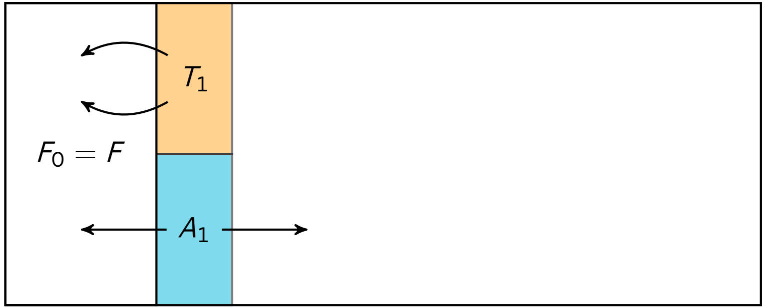
Sketch of the proof for reachability games
A: Attractor
T: Trap
Arena
$F_0 = F$

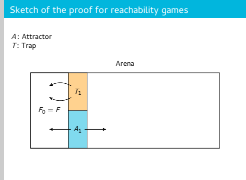
The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena



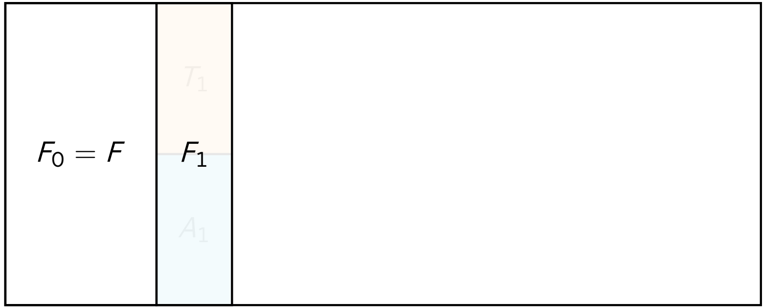


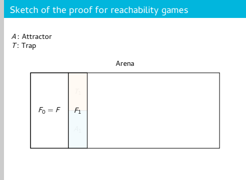
The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena



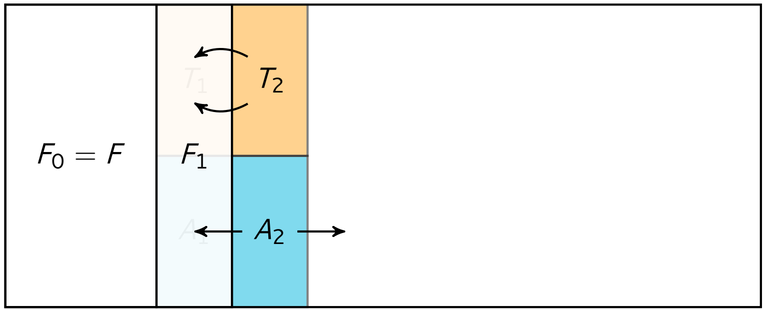


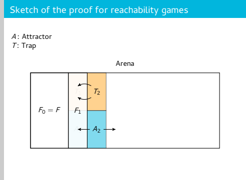
The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena



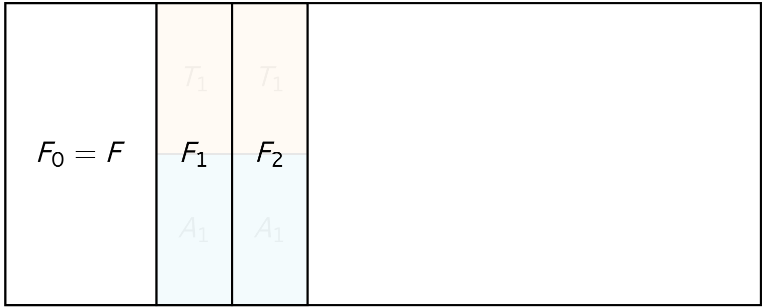


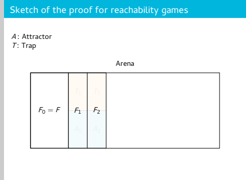
The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena



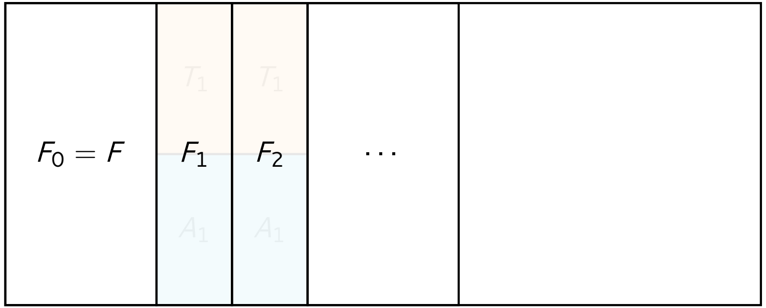


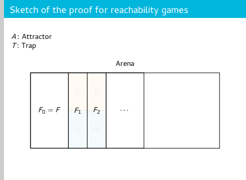
The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena



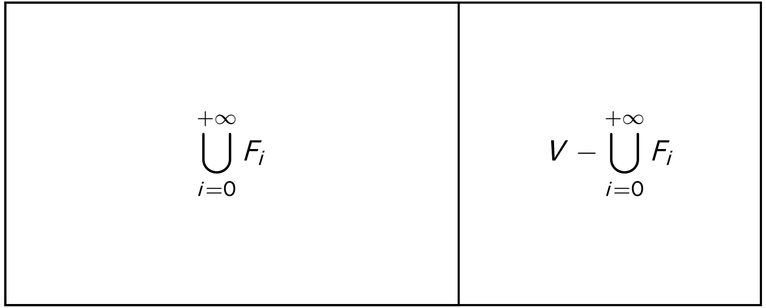


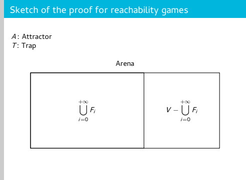
The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

Arena





The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

Sketch of the proof for reachability games

A: Attractor
T: Trap

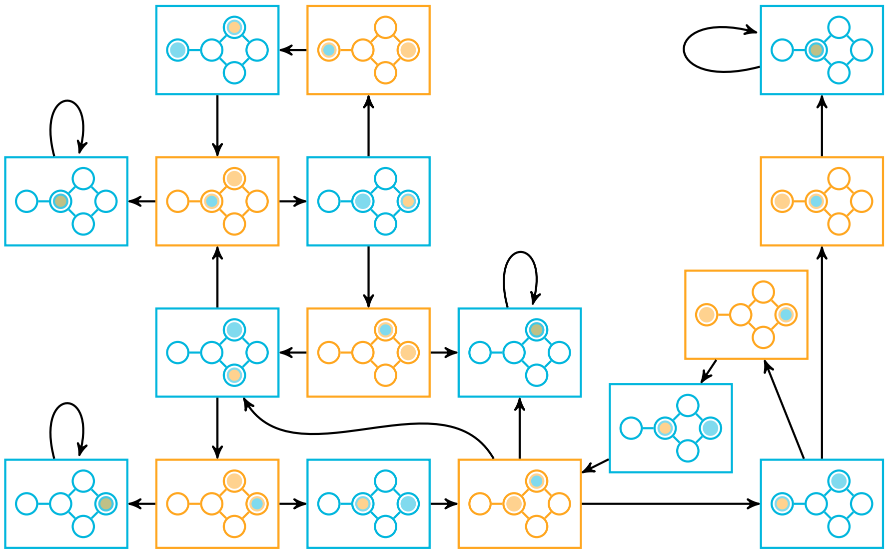
Arena

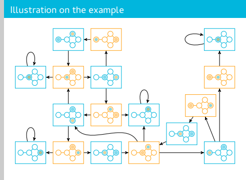
$\bigcup_{i=0}^{+\infty} F_i$ Winning region for Eve	$V - \bigcup_{i=0}^{+\infty} F_i$ Winning region for Adam

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

The proof goes as follows. Let's consider the set of final states $F = F_0$ for Eve. Now let's consider the different states that lead to F. If Eve was playing, she was in a set of states that we call A, as Attractor, in which for sure Eve will move towards a final state in F (because Eve wants to win). Otherwise, Adam was playing, and he didn't have the choice but move towards a final state for Eve (we call this situation a trap). If we make the union of this attractor and trap, we can consider it as a new set of final states for Eve, F_1. In order to end of in F_1, the same reasoning can be done, defining an attractor and a trap. This can be done recursively until we can define the winning regions for both players.

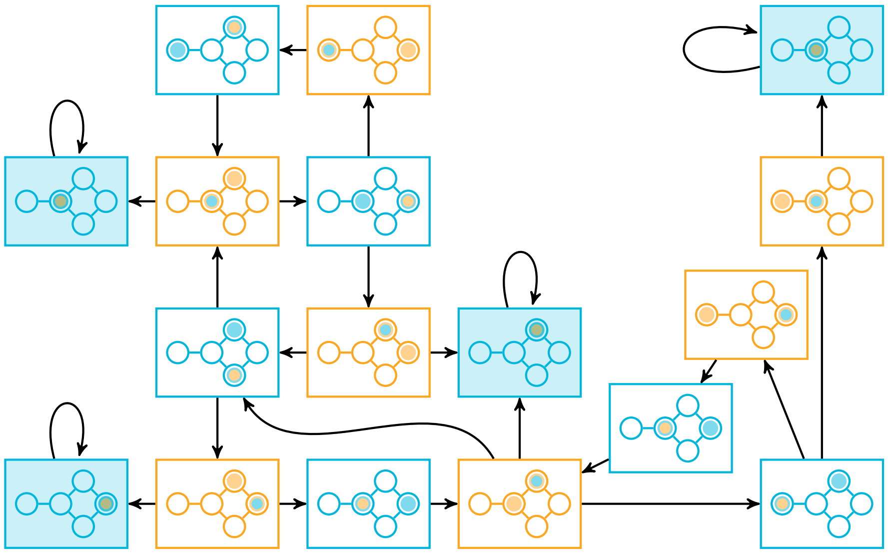
Illustration on the example

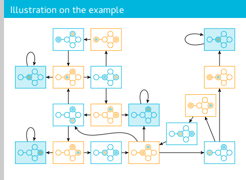




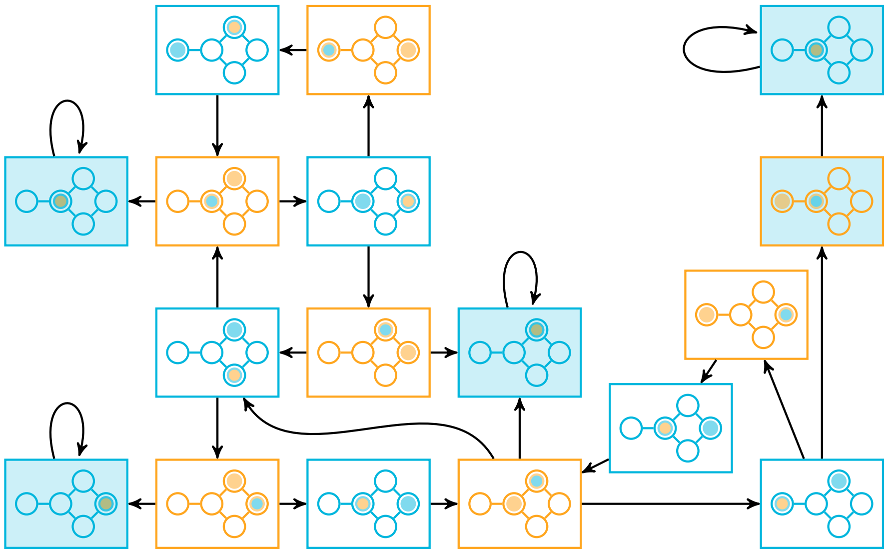
Let's illustrate this with our cop and robber. In blue we show the final states (Eve wins, the cop catches the robber). Then for example on the right, we show that there was no other choice for the robber but to go be caught by the cop. So this vertex is part of the winning region for Eve, too. However, all these other situations in which the robber is at 2 hops from the cops are the winning region for Adam, because Adam (the robber) can always choose to move in the opposite direction, then it is never caught.

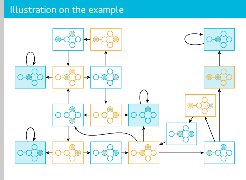
Illustration on the example





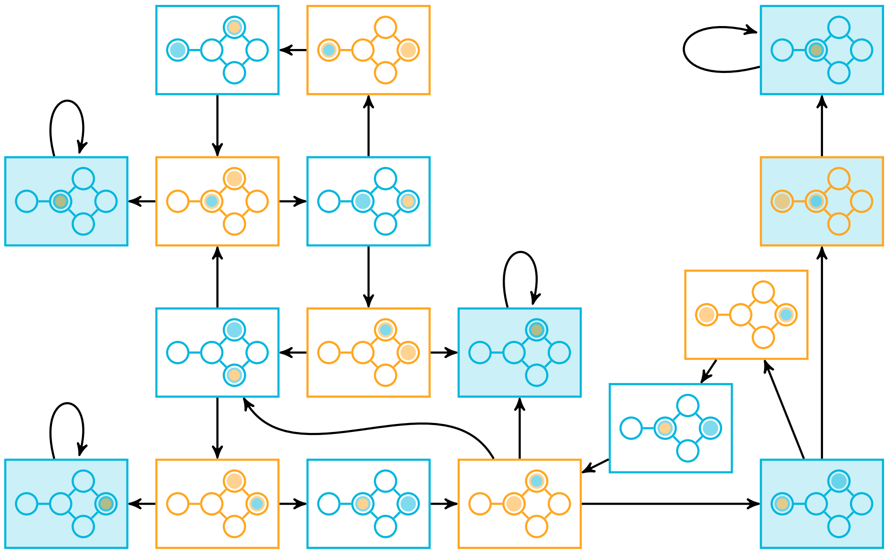
Let's illustrate this with our cop and robber. In blue we show the final states (Eve wins, the cop catches the robber). Then for example on the right, we show that there was no other choice for the robber but to go be caught by the cop. So this vertex is part of the winning region for Eve, too. However, all these other situations in which the robber is at 2 hops from the cops are the winning region for Adam, because Adam (the robber) can always choose to move in the opposite direction, then it is never caught.

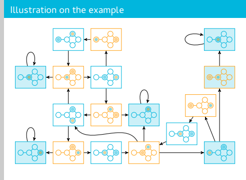




Let’s illustrate this with our cop and robber. In blue we show the final states (Eve wins, the cop catches the robber). Then for example on the right, we show that there was no other choice for the robber but to go be caught by the cop. So this vertex is part of the winning region for Eve, too. However, all these other situations in which the robber is at 2 hops from the cops are the winning region for Adam, because Adam (the robber) can always choose to move in the opposite direction, then it is never caught.

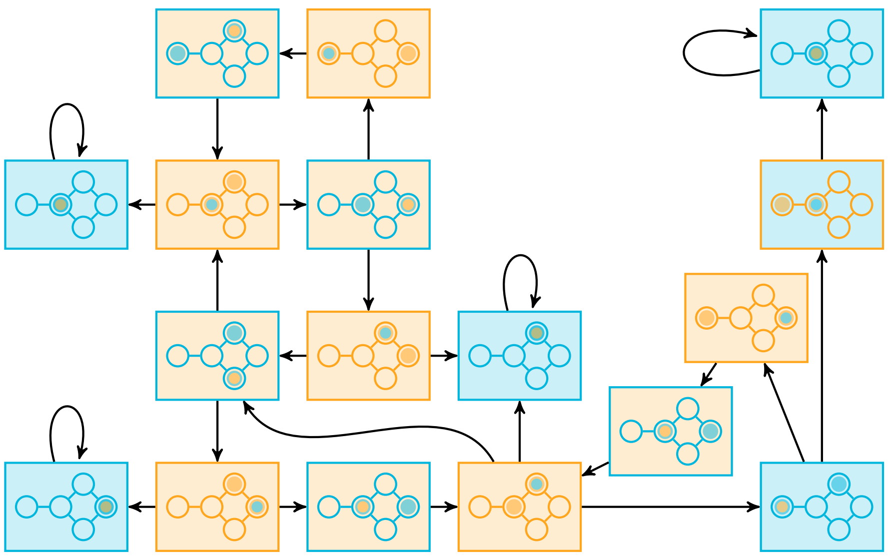
Illustration on the example

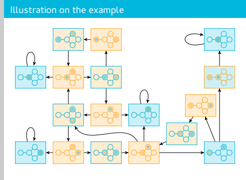




Let's illustrate this with our cop and robber. In blue we show the final states (Eve wins, the cop catches the robber). Then for example on the right, we show that there was no other choice for the robber but to go be caught by the cop. So this vertex is part of the winning region for Eve, too. However, all these other situations in which the robber is at 2 hops from the cops are the winning region for Adam, because Adam (the robber) can always choose to move in the opposite direction, then it is never caught.

Illustration on the example

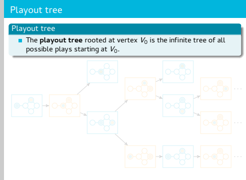




Let's illustrate this with our cop and robber. In blue we show the final states (Eve wins, the cop catches the robber). Then for example on the right, we show that there was no other choice for the robber but to go be caught by the cop. So this vertex is part of the winning region for Eve, too. However, all these other situations in which the robber is at 2 hops from the cops are the winning region for Adam, because Adam (the robber) can always choose to move in the opposite direction, then it is never caught.

Playout tree

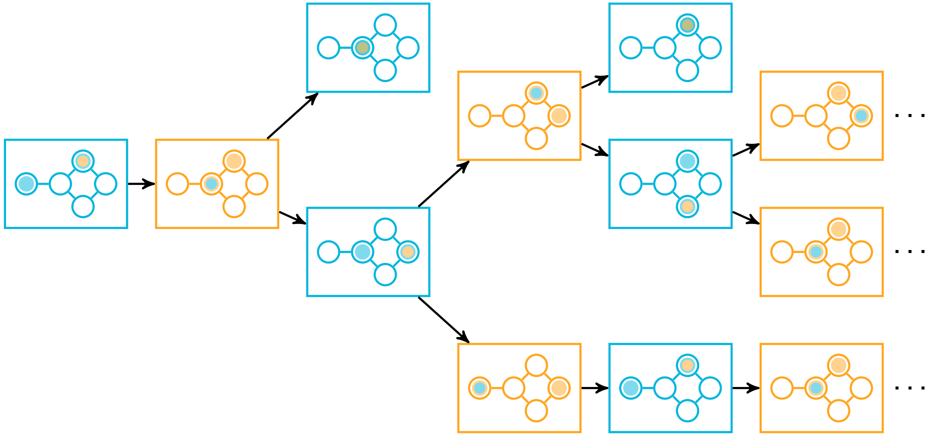
- The **playout tree** rooted at vertex V_0 is the infinite tree of all possible plays starting at V_0.

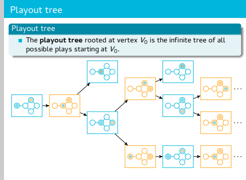


This is only at this slide that we can make the link with machine learning. As tree search is most of the time computationally infeasible, research is actively being done on how to apply machine learning to explore large trees in this context.

Playout tree

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





This is only at this slide that we can make the link with machine learning. As tree search is most of the time computationally infeasible, research is actively being done on how to apply machine learning to explore large trees in this context.

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.

This is only at this slide that we can make the link with machine learning. As tree search is most of the time computationally infeasible, research is actively being done on how to apply machine learning to explore large trees in this context.

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.

This is only at this slide that we can make the link with machine learning. As tree search is most of the time computationally infeasible, research is actively being done on how to apply machine learning to explore large trees in this context.

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

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