Game-Theoretic Approach to Temporal Synthesis

Introduction

Antonio Di Stasio **Giuseppe Perelli** [1] Shufang Zhu

2nd European Summer School on Artificial Intelligence
Athens (Greece) 15-19 July 2024

[1]The realization of this course was partially supported by MUR under the PRIN programme, grant B87G22000450001 (PINPOINT).

SAPIENZA
UNIVERSITÀ DI ROMA

Introduction to Games, Temporal Logic specifications	Giuseppe
Automata-Theoretic Approach, Synthesis	Giuseppe
LTLf Synthesis under Environment Specifications	Antonio
Notable Cases of LTLf Synthesis under LTL Environment Specifications	Shufang
Symbolic Synthesis	Shufang

Related courses at ESSAI

11:00-12:30 - Formal Aspects of Strategic Reasoning and Game Playing

11:00-12:30 - Logic-Based Specification and Verification of Multi-Agent Systems

Assistant Professor @ Sapienza University of Rome

Ph.D. in Computer Science (Background in Mathematics)

Main research interests:

- Formal Methods for Artificial Intelligence
- Logics and Games for Multi-Agent Systems
- Synthesis and Rational Synthesis

Website https://giuseppeperelli.github.io
Email: perelli@di.uniroma1.it

SCAN ME

Senior Research Associate @ University of Oxford

Soon to join University of Liverpool as Lecturer ... Congrats!

Main research interests: interdisciplinary knowledge across artificial intelligence (AI) and formal methods (FM)

- Automated Reasoning
- Planning
- Synthesis

Website https://shufang-zhu.github.io/
Email: shufang.zhu@cs.ox.ac.uk

Senior Research Associate @ University of Oxford

Soon to join City University of London as Lecturer ... Congrats!

Main research interests:

- Game Theory
- Parity Games
- Formal Aspects of System Specification, Verification, Synthesis
- Automated Planning

Website https://antoniodistasio.github.io/
Email: antonio.distasio@cs.ox.ac.uk

Agents are powerful models in many areas of Computer Science.

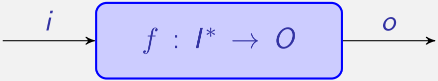
Three characteristics

- Capabilities: actions and constraints
- Knowledge: information about environment
- Goal: specification of a task/objective to fulfill

Appears in many areas

Robotics
Software Engineering
Process Management
Knowledge Representation
Planning
Multi-Agent Systems
Sequential decision making
Reinforcement learning

Process



Function f sends outputs according to the history of inputs.

Abadi, Lamport, Wolper - Realizable and Unrealizable Specifications of Reactive Systems. - ICALP'89

– Adhere to capabilities: actions always fulfill constraints
– Depend on knowledge: react on the stream of inputs
– Fulfill the specification

An agent satisfying these properties is correct.

Temporal specification setting

$$f \rightsquigarrow \mathcal{T}_f = \langle Q, I, O, \delta, \tau \rangle$$

Finite-state machines are expressive enough to implement agents correctly in a large class of temporal specifications.

Instead of writing programs, we write specifications and run an automatic synthesis procedure that in turns produces the program.

Reactive Synthesis

- Self-programming mechanism.
- Specifying a problem is usually simpler than solving it.
- Aim: correct-by-construction.

Pnueli and Rosner - On the Synthesis of a Reactive Module. - POPL'89

Finkbeiner - Synthesis of Reactive Systems. - DSSE'16

Synthesis problems as games

Agent vs environment	$\Longleftrightarrow$	Two-Player Game
Temporal specification	$\Longleftrightarrow$	Winning Condition
Correct program	$\Longleftrightarrow$	Winning Strategy

Solving synthesis = winning a game

Synthesizing a correct program reduces to winning a suitably defined formal game.
Solution techniques: Logic, Games, and Automata.

SAPIENZA UNIVERSITÀ DI ROMA

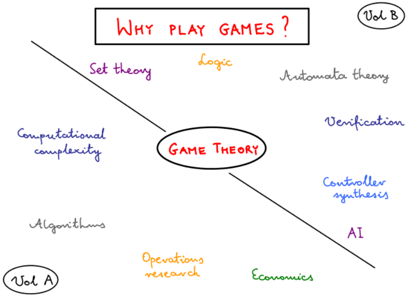
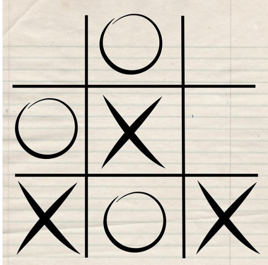
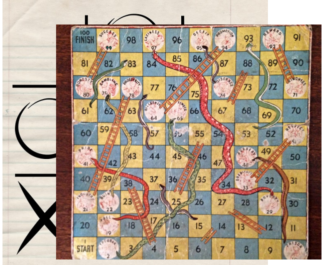


Image credits: Martin Zimmerman

- It's fun!
- Model reactive systems
- Solve synthesis problems
- Evaluate logic formulas




100
FINISH
1
START


100
FINISH
START


100
FINISH
98
96
93
91
81
61
60
40
20
START

Synthesis

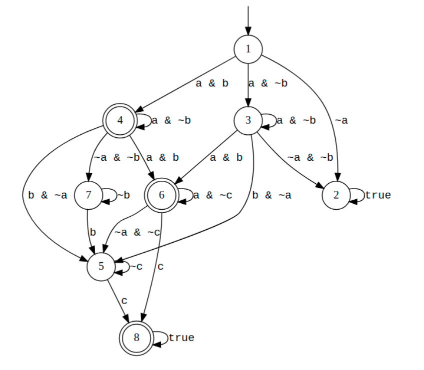


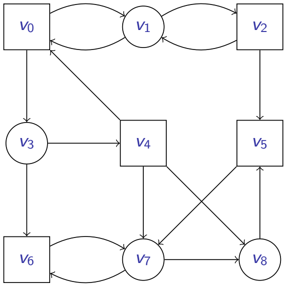
Image credits: ltlf2dfa.diag.uniroma1.it

SAPIENZA UNIVERSITÀ DI ROMA

- Players
  - 1 player;
  - 2 players;
  - multi-players.
- Interaction
  - Turn-based;
  - Concurrent.
- Information
  - Perfect;
  - Imperfect.
- Nature
  - Deterministic;
  - Stochastic.
- Objective
  - Reachability;
  - Safety;
  - Büchi;
  - co-Büchi;
  - Parity, Rabin, Streett, Muller, . . .

Today

2-player turn-based perfect information games.

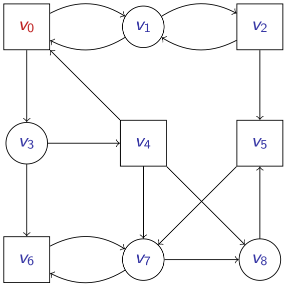


A Game is played over a (finite) graph (V, E), whose vertexes are under the control of the two players $\mathrm{V} = \mathrm{V}_0 \cup \mathrm{V}_1$.

A token moves along the vertexes and sent to a successor by the controlling player.

The outcome or play is an infinite sequence of vertexes in the graph.

A winning condition/objective is a subset $\mathsf{Obj} \subseteq \mathrm{V}^{\omega}$ of plays that Player 0 wants to occur.



A Game is played over a (finite) graph (V, E), whose vertexes are under the control of the two players $\mathrm{V} = \mathrm{V}_0 \cup \mathrm{V}_1$.

A token moves along the vertexes and sent to a successor by the controlling player.

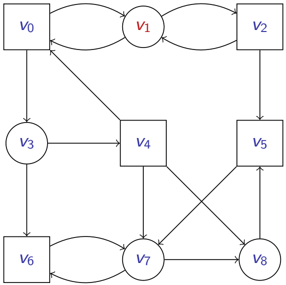
The outcome or play is an infinite sequence of vertexes in the graph.

A winning condition/objective is a subset $\mathrm{Obj} \subseteq \mathrm{V}^\omega$ of plays that Player 0 wants to occur.

Sample play

$\pi = v_0$

SAPIENZA UNIVERSITÀ DI ROMA



A Game is played over a (finite) graph (V, E), whose vertexes are under the control of the two players $\mathrm{V} = \mathrm{V}_0 \cup \mathrm{V}_1$.

A token moves along the vertexes and sent to a successor by the controlling player.

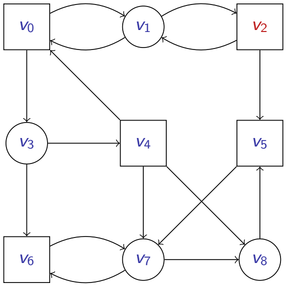
The outcome or play is an infinite sequence of vertexes in the graph.

A winning condition/objective is a subset $\mathsf{Obj} \subseteq \mathrm{V}^\omega$ of plays that Player 0 wants to occur.

Sample play

$\pi = v_0 \cdot v_1$

SAPIENZA UNIVERSITÀ DI ROMA



A Game is played over a (finite) graph (V, E), whose vertexes are under the control of the two players $\mathrm{V} = \mathrm{V}_0 \cup \mathrm{V}_1$.

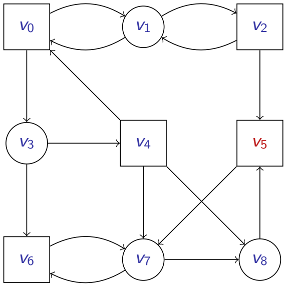
A token moves along the vertexes and sent to a successor by the controlling player.

The outcome or play is an infinite sequence of vertexes in the graph.

A winning condition/objective is a subset $\mathrm{Obj} \subseteq \mathrm{V}^\omega$ of plays that Player 0 wants to occur.

Sample play

$\pi = v_0 \cdot v_1 \cdot v_2$



A Game is played over a (finite) graph (V, E), whose vertexes are under the control of the two players $\mathrm{V} = \mathrm{V}_0 \cup \mathrm{V}_1$.

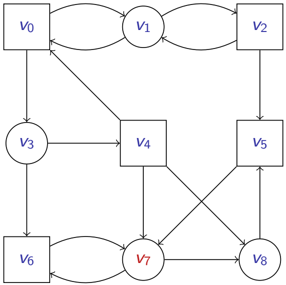
A token moves along the vertexes and sent to a successor by the controlling player.

The outcome or play is an infinite sequence of vertexes in the graph.

A winning condition/objective is a subset $\mathsf{Obj} \subseteq \mathrm{V}^\omega$ of plays that Player 0 wants to occur.

Sample play

$\pi = v_0 \cdot v_1 \cdot v_2 \cdot v_5$



A Game is played over a (finite) graph (V, E), whose vertexes are under the control of the two players $\mathrm{V} = \mathrm{V}_0 \cup \mathrm{V}_1$.

A token moves along the vertexes and sent to a successor by the controlling player.

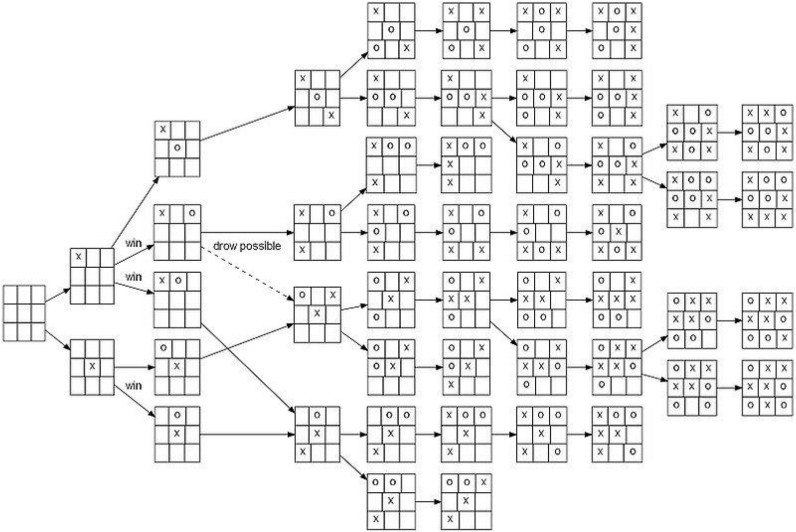
The outcome or play is an infinite sequence of vertexes in the graph.

A winning condition/objective is a subset $\mathsf{Obj} \subseteq \mathrm{V}^\omega$ of plays that Player 0 wants to occur.

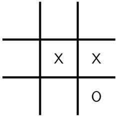
Sample play

$\pi = v_0 \cdot v_1 \cdot v_2 \cdot v_5 \cdot v_7 \cdot \ldots \in \mathrm{V}^\omega$

Modelling tic tac toe

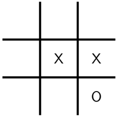


Tic-Tac-Toe is played on a 3×3 grid. Two players place their placeholders in turn on a free square. The first to place three of its own placeholders aligned wins.



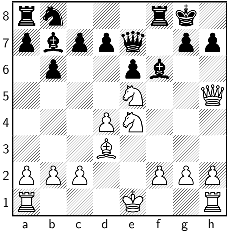
vertexes $\approx 9! \sim 10^5$

Tic-Tac-Toe is played on a 3×3 grid. Two players place their placeholders in turn on a free square. The first to place three of its own placeholders aligned wins.



vertexes $\approx 9! \sim 10^5$

Lasker vs Thomas 1912: White to move and mate in 7



vertexes $\approx 10^{43} - 10^{50}$ (Shannon, 1950)

edges $\approx 10^{123}$ (Allis, 1994)

possible different games $\approx 10^{10^{50}}$

Size of 5-pieces tablebase: 7GB

Size of 6-pieces tablebase: 1,2TB

Size of 7-pieces tablebase: 140TB (“Deep Thinking”, Kasparov, 2017)

Type of objectives

For a subset of the vertexes $T \subseteq \mathrm{V}$:

SAPIENZA UNIVERSITÀ DI ROMA

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

SAPIENZA
UNIVERSITÀ DI ROMA

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\mathsf{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\mathsf{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\mathsf{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

$$\mathsf{Safe}(T) = \{\pi \in \mathrm{V}^\omega : \forall i \in \mathbb{N}, \pi[i] \in T\}$$

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\text{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

$$\text{Safe}(T) = \{\pi \in \mathrm{V}^\omega : \forall i \in \mathbb{N}, \pi[i] \in T\}$$

- Büchi: visit T infinitely often.

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\mathsf{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

$$\mathsf{Safe}(T) = \{\pi \in \mathrm{V}^\omega : \forall i \in \mathbb{N}, \pi[i] \in T\}$$

- Büchi: visit T infinitely often.

$$\mathsf{Buchi}(T) = \{\pi \in \mathrm{V}^\omega : \forall j \in \mathbb{N}, \exists i > j, \pi[i] \in T\}$$

SAPIENZA UNIVERSITÀ DI ROMA

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\text{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

$$\text{Safe}(T) = \{\pi \in \mathrm{V}^\omega : \forall i \in \mathbb{N}, \pi[i] \in T\}$$

- Büchi: visit T infinitely often.

$$\text{Buchi}(T) = \{\pi \in \mathrm{V}^\omega : \forall j \in \mathbb{N}, \exists i > j, \pi[i] \in T\}$$

- co-Büchi: reach and stay in T forever.

SAPIENZA UNIVERSITÀ DI ROMA

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\mathrm{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

$$\mathrm{Safe}(T) = \{\pi \in \mathrm{V}^\omega : \forall i \in \mathbb{N}, \pi[i] \in T\}$$

- Büchi: visit T infinitely often.

$$\mathrm{Buchi}(T) = \{\pi \in \mathrm{V}^\omega : \forall j \in \mathbb{N}, \exists i > j, \pi[i] \in T\}$$

- co-Büchi: reach and stay in T forever.

$$\mathrm{coBuchi}(T) = \{\pi \in \mathrm{V}^\omega : \exists j \in \mathbb{N}, \forall i > j, \pi[i] \in T\}$$

Type of objectives

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\mathrm{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

$$\mathrm{Safe}(T) = \{\pi \in \mathrm{V}^\omega : \forall i \in \mathbb{N}, \pi[i] \in T\}$$

- Büchi: visit T infinitely often.

$$\mathrm{Buchi}(T) = \{\pi \in \mathrm{V}^\omega : \forall j \in \mathbb{N}, \exists i > j, \pi[i] \in T\}$$

- co-Büchi: reach and stay in T forever.

$$\mathrm{coBuchi}(T) = \{\pi \in \mathrm{V}^\omega : \exists j \in \mathbb{N}, \forall i > j, \pi[i] \in T\}$$

- Parity, Rabin, Streett, Muller, LTL, ...

For a subset of the vertexes $T \subseteq \mathrm{V}$:

- Reachability: visit T at least once.

$$\mathsf{Reach}(T) = \{\pi \in \mathrm{V}^\omega : \exists i \in \mathbb{N}, \pi[i] \in T\}$$

- Safety: stay in T forever.

$$\mathsf{Safe}(T) = \{\pi \in \mathrm{V}^\omega : \forall i \in \mathbb{N}, \pi[i] \in T\}$$

- Büchi: visit T infinitely often.

$$\mathsf{Buchi}(T) = \{\pi \in \mathrm{V}^\omega : \forall j \in \mathbb{N}, \exists i > j, \pi[i] \in T\}$$

- co-Büchi: reach and stay in T forever.

$$\mathsf{coBuchi}(T) = \{\pi \in \mathrm{V}^\omega : \exists j \in \mathbb{N}, \forall i > j, \pi[i] \in T\}$$

- Parity, Rabin, Streett, Muller, LTL, ...

Question: what if we have more "alternations" of existential and universal quantifiers?

SAPIENZA UNIVERSITÀ DI ROMA

Strategies

A strategy maps partial outcomes (i.e., finite sequences of vertexes) into successors and it is of the form

- $\sigma_0 : \mathrm{V}^* \cdot \mathrm{V}_0 \to \mathrm{V}$ Player 0 strategy
- $\sigma_1 : \mathrm{V}^* \cdot \mathrm{V}_1 \to \mathrm{V}$ Player 1 strategy

Consistent plays

Strategies "restricts" the game only to those plays π that are consistent with σ_0, that is such that:

$$\pi[i+1] = \sigma_0(\pi[0] \cdot \pi[1] \cdot \ldots \cdot \pi[i])$$

For each σ_0, σ_1, there is only one consistent play $\pi(v, \sigma_0, \sigma_1)$ starting from v.

Winning strategies

A strategy σ_0 is winning for Player 0 in v if every consistent path π starting from v belongs to Obj. (Winning set $\text{Win}_0 \subseteq V$)

A strategy σ_1 is winning for Player 1 in v if every consistent path π starting from v does not belong to Obj. (Losing set $\text{Win}_1 \subseteq V$)

Solving a game

The solution of a game $\mathcal{G}$ is the set Win_0 of vertexes that are winning for Player 0, altogether with a winning strategy σ_0.

SAPIENZA UNIVERSITÀ DI ROMA

Winning strategies

A strategy σ_0 is winning for Player 0 in v if every consistent path π starting from v belongs to Obj. (Winning set $\mathrm{Win}_0 \subseteq V$)

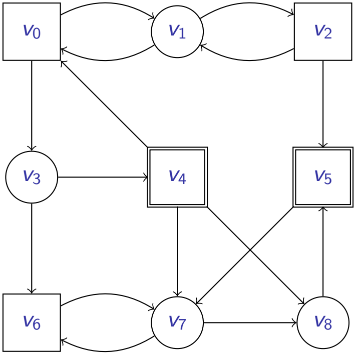
A strategy σ_1 is winning for Player 1 in v if every consistent path π starting from v does not belong to Obj. (Losing set $\mathrm{Win}_1 \subseteq V$)

Solving a game

The solution of a game $\mathcal{G}$ is the set Win_0 of vertexes that are winning for Player 0, altogether with a winning strategy σ_0.

Warning! While $\mathrm{Win}_0 \cap \mathrm{Win}_1 = \emptyset$, it is not always the case that $\mathrm{V} = \mathrm{Win}_0 \cup \mathrm{Win}_1$.

Consider again the arena below and let $T = \{v_4, v_5\}$ (the double bordered nodes).



What is the winning set of $\mathcal{G}$?

Consider the function force_0 defined as follows:

$$\text{force}_0(X) = \{v \in \mathrm{V}_0 : E(v) \cap X \neq \emptyset\} \cup \{v \in \mathrm{V}_1 : E(v) \subseteq X\}$$

Player 0 has a move to enter the region X;

Player 1 cannot avoid to enter the region X.

The function computes the vertexes from which Player 0 can enforce the token to move in the subset X of vertexes.

SAPIENZA UNIVERSITÀ DI ROMA

Constrained problem

$\text{Reach}^{n}(T) :=$ "Player 0 can reach T in at most n moves".

Constrained problem

$\text{Reach}^n(T) :=$ "Player 0 can reach T in at most n moves".

$n = 0$: T I have to be in T already.

SAPIENZA UNIVERSITÀ DI ROMA

Constrained problem

$\text{Reach}^n(T) :=$ "Player 0 can reach T in at most n moves".

$n = 0$: T I have to be in T already.

$$\text{Reach}^0(T) = T$$

Constrained problem

$\text{Reach}^n(T) :=$ "Player 0 can reach T in at most n moves".

$n = 0$: T I have to be in T already.

$\text{Reach}^0(T) = T$

$n > 0$: either I am or can force to a vertex winning in at most $n - 1$ moves.

SAPIENZA UNIVERSITÀ DI ROMA

Constrained problem

$\text{Reach}^n(T) :=$ "Player 0 can reach T in at most n moves".

$n = 0$: T I have to be in T already.

$$\text{Reach}^0(T) = T$$

$n > 0$: either I am or can force to a vertex winning in at most $n - 1$ moves.

$$\text{Reach}^n(T) = \text{Reach}^{n-1}(T) \cup \text{force}_0(\text{Reach}^{n-1}(T))$$

SAPIENZA UNIVERSITÀ DI ROMA

Constrained problem

$\mathrm{Reach}^n(T) :=$ "Player 0 can reach T in at most n moves".

$n = 0$: T I have to be in T already.

$$\mathrm{Reach}^0(T) = T$$

$n > 0$: either I am or can force to a vertex winning in at most $n - 1$ moves.

$$\mathrm{Reach}^n(T) = \mathrm{Reach}^{n-1}(T) \cup \mathrm{force}_0(\mathrm{Reach}^{n-1}(T))$$

Solving reachability

$\mathrm{Win}_0(\mathcal{G}) = \mathrm{Reach}(T) :=$ "Player 0 can reach T ~~in at most n moves~~".

Sapienza Università di Roma

Constrained problem

$$\text{Reach}^n(T) := \text{"Player 0 can reach } T \text{ in at most } n \text{ moves"}.$$

$n = 0$: T I have to be in T already.

$$\text{Reach}^0(T) = T$$

$n > 0$: either I am or can force to a vertex winning in at most $n - 1$ moves.

$$\text{Reach}^n(T) = \text{Reach}^{n-1}(T) \cup \text{force}_0(\text{Reach}^{n-1}(T))$$

Solving reachability

$$\text{Win}_0(\mathcal{G}) = \text{Reach}(T) := \text{"Player 0 can reach } T \text{ ~~in at most } n \text{ moves~~"}.$$

$$\text{Reach}(T) = \text{Reach}^0(T) \cup \text{Reach}^1(T) \cup \ldots \cup \text{Reach}^n(T) \cup \ldots$$

SAPIENZA UNIVERSITÀ DI ROMA

Constrained problem

$\text{Reach}^n(T) :=$ "Player 0 can reach T in at most n moves".

$n = 0$: T I have to be in T already.

$$\text{Reach}^0(T) = T$$

$n > 0$: either I am or can force to a vertex winning in at most $n-1$ moves.

$$\text{Reach}^n(T) = \text{Reach}^{n-1}(T) \cup \text{force}_0(\text{Reach}^{n-1}(T))$$

Solving reachability

$\text{Win}_0(\mathcal{G}) = \text{Reach}(T) :=$ "Player 0 can reach T ~~in at most n moves~~".

$$\text{Reach}(T) = \text{Reach}^0(T) \cup \text{Reach}^1(T) \cup \ldots \cup \text{Reach}^n(T) \cup \ldots$$

$$\text{Reach}^0(T) \subseteq \text{Reach}^1(T) \subseteq \ldots \subseteq \text{Reach}^n(T) \subseteq \ldots$$

SAPIENZA UNIVERSITÀ DI ROMA

Constrained problem

$\mathrm{Reach}^n(T) :=$ "Player 0 can reach T in at most n moves".

$n = 0$: T I have to be in T already.

$$\mathrm{Reach}^0(T) = T$$

$n > 0$: either I am or can force to a vertex winning in at most $n-1$ moves.

$$\mathrm{Reach}^n(T) = \mathrm{Reach}^{n-1}(T) \cup \mathrm{force}_0(\mathrm{Reach}^{n-1}(T))$$

Solving reachability

$\mathrm{Win}_0(\mathcal{G}) = \mathrm{Reach}(T) :=$ "Player 0 can reach T ~~in at most n moves~~".

$$\mathrm{Reach}(T) = \mathrm{Reach}^0(T) \cup \mathrm{Reach}^1(T) \cup \ldots \cup \mathrm{Reach}^n(T) \cup \ldots$$

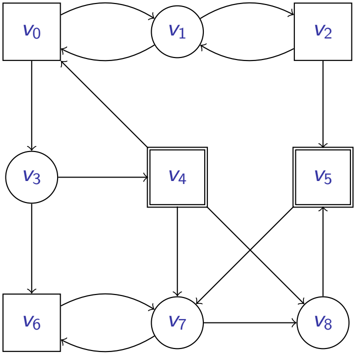
$$\mathrm{Reach}^0(T) \subseteq \mathrm{Reach}^1(T) \subseteq \ldots \subseteq \mathrm{Reach}^n(T) \subseteq \ldots$$

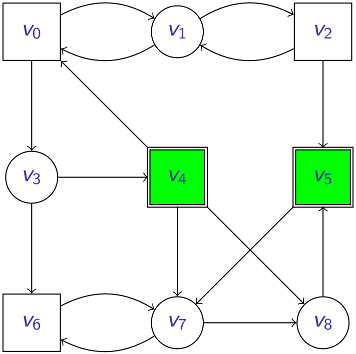
Fix-point calculation $\mu\mathcal{X}.(T \cup \mathrm{force}_0(\mathcal{X}))$

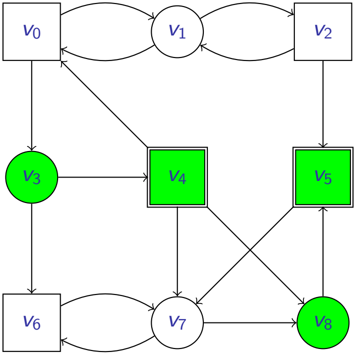
SAPIENZA UNIVERSITÀ DI ROMA

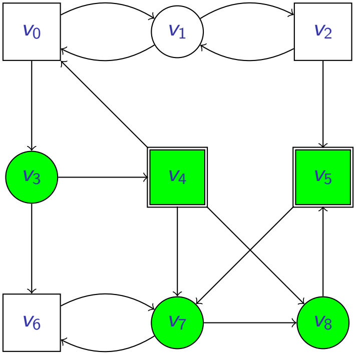
Algorithm 1 Reachability game

1: $\mathsf{Win}_{old} := T$
2: $\mathsf{Win} := \mathsf{Win}_{old} \cup \mathsf{force}_{\mathrm{o}}(\mathsf{Win}_{old})$
3: **while** $\mathsf{Win} \neq \mathsf{Win}_{old}$ **do**
4: $\quad \mathsf{Win}_{old} := \mathsf{Win}$
5: $\quad \mathsf{Win} := \mathsf{Win} \cup \mathsf{force}_{\mathrm{o}}(\mathsf{Win})$
6: **end while**
7: **return** Win

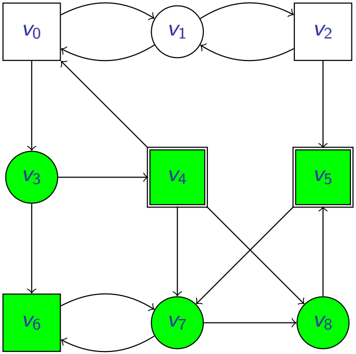

v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8


v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8


v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8


v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8

Sample Execution



SAPIENZA UNIVERSITÀ DI ROMA

Memoryless strategy

A strategy σ_0 is memoryless if it is of the form

$$\sigma_0 : \cancel{V^* \cdot} V_0 \to V$$

that is, at every vertex v, the next move does not depend on the past history (and thus it is always the same).

SAPIENZA UNIVERSITÀ DI ROMA

Memoryless strategy

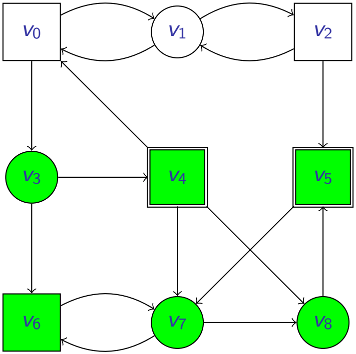
A strategy σ_0 is memoryless if it is of the form

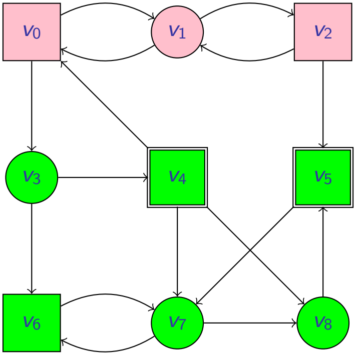
$$\sigma_0 : \cancel{V^* \cdot} V_o \to V$$

that is, at every vertex v, the next move does not depend on the past history (and thus it is always the same).

Theorem (Memoryless)

If $v \in \mathrm{Win}_o$, then there exists a memoryless strategy σ_0 that is winning from v.

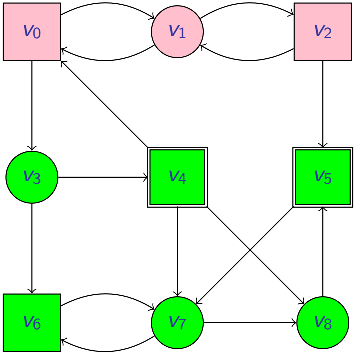

v0
v1
v2
v3
v4
v5
v6
v7
v8



It holds that $\text{Win}_0 \cup \text{Win}_1 = V$. When this is the case, we say that the game is determined.

Theorem (determinacy)

Every 2-player turn-based reachability game is determined.



It holds that $\mathrm{Win}_0 \cup \mathrm{Win}_1 = V$. When this is the case, we say that the game is determined.

Theorem (determinacy)

Every 2-player turn-based reachability game is determined.

Consider an arena $\mathbf{A} = (\mathrm{V}, E, \mathrm{V}_0, \mathrm{V}_1)$ and a safety game $\mathcal{G} = (\mathbf{A}, \mathsf{Safe}(T))$.
Define the dual arena $\overline{\mathbf{A}} = (\mathrm{V}, E, \mathrm{V}_1, \mathrm{V}_0)$ and the reachability game $\overline{\mathcal{G}} = (\overline{\mathbf{A}}, \mathsf{Reach}(\mathrm{V} \setminus T))$

Exercise - Prove that:

$\mathsf{Win}_0(\mathcal{G}) = \mathsf{Win}_1(\overline{\mathcal{G}})$;

$\mathsf{Win}_1(\mathcal{G}) = \mathsf{Win}_0(\overline{\mathcal{G}})$.

Theorem

We can solve safety games by solving the dual reachability game and complement the solution.

Problem

$\mathsf{Safe}^n(T) :=$ "Player 0 can stay in T for at least n moves."

Problem

$\text{Safe}^n(T)$:= "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already.

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\text{Safe}^n(T) :=$ "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already. $\text{Safe}^0(T) = T$

Problem

$\mathsf{Safe}^n(T) :=$ "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already. $\mathsf{Safe}^0(T) = T$

For $n > 0$: I must stay in T and move to a vertex from which I can force to stay in T for $n - 1$ more times.

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\mathsf{Safe}^n(T) :=$ "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already. $\mathsf{Safe}^0(T) = T$

For $n > 0$: I must stay in T and move to a vertex from which I can force to stay in T for $n - 1$ more times. $\mathsf{Safe}^n(T) = T \cap \mathsf{force}_0(\mathsf{Safe}^{n-1}(T))$

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\mathrm{Safe}^n(T) :=$ "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already. $\mathrm{Safe}^0(T) = T$

For $n > 0$: I must stay in T and move to a vertex from which I can force to stay in T for $n - 1$ more times. $\mathrm{Safe}^n(T) = T \cap \mathrm{force}_0(\mathrm{Safe}^{n-1}(T))$

Solving safety

$\mathrm{Win}_0(\mathcal{G}) = \mathrm{Safe}(T) :=$ "Player 0 can stay in T forever".

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\text{Safe}^n(T)$:= "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already. $\text{Safe}^0(T) = T$

For $n > 0$: I must stay in T and move to a vertex from which I can force to stay in T for $n - 1$ more times. $\text{Safe}^n(T) = T \cap \text{force}_0(\text{Safe}^{n-1}(T))$

Solving safety

$\text{Win}_0(\mathcal{G}) = \text{Safe}(T)$:= "Player 0 can stay in T forever".

$$\text{Win}_0(\mathcal{G}) = \text{Safe}(T) = \text{Safe}^0(T) \cap \text{Safe}^1(T) \cap \ldots \cap \text{Safe}^n(T) \cap \ldots$$

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\mathrm{Safe}^n(T)$:= "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already. $\mathrm{Safe}^0(T) = T$

For $n > 0$: I must stay in T and move to a vertex from which I can force to stay in T for $n - 1$ more times. $\mathrm{Safe}^n(T) = T \cap \mathrm{force}_0(\mathrm{Safe}^{n-1}(T))$

Solving safety

$\mathrm{Win}_0(\mathcal{G}) = \mathrm{Safe}(T)$:= "Player 0 can stay in T forever".

$$\mathrm{Win}_0(\mathcal{G}) = \mathrm{Safe}(T) = \mathrm{Safe}^0(T) \cap \mathrm{Safe}^1(T) \cap \ldots \cap \mathrm{Safe}^n(T) \cap \ldots$$

$$\mathrm{Safe}^0(T) \supseteq \mathrm{Safe}^1(T) \supseteq \ldots \supseteq \mathrm{Safe}^n(T) \supseteq \ldots$$

Problem

$\text{Safe}^n(T) :=$ "Player 0 can stay in T for at least n moves."

For $n = 0$: I have to be in T already. $\text{Safe}^0(T) = T$

For $n > 0$: I must stay in T and move to a vertex from which I can force to stay in T for $n - 1$ more times. $\text{Safe}^n(T) = T \cap \text{force}_0(\text{Safe}^{n-1}(T))$

Solving safety

$\text{Win}_0(\mathcal{G}) = \text{Safe}(T) :=$ "Player 0 can stay in T forever".

$$\text{Win}_0(\mathcal{G}) = \text{Safe}(T) = \text{Safe}^0(T) \cap \text{Safe}^1(T) \cap \ldots \cap \text{Safe}^n(T) \cap \ldots$$

$$\text{Safe}^0(T) \supseteq \text{Safe}^1(T) \supseteq \ldots \supseteq \text{Safe}^n(T) \supseteq \ldots$$

Fix-point calculation $\nu \mathcal{Y}.(T \cap \text{force}_0(\mathcal{Y}))$

Algorithm 2 Safety game

$\mathsf{Win}_{old} := T$
$\mathsf{Win} := \mathsf{Win}_{old} \cap \mathsf{force}_{o}(\mathsf{Win}_{old})$
while $\mathsf{Win} \neq \mathsf{Win}_{old}$ **do**
    $\mathsf{Win}_{old} := \mathsf{Win}$
    $\mathsf{Win} := \mathsf{Win} \cap \mathsf{force}_{o}(\mathsf{Win})$
end while
return Win

Question: How do we solve Büchi and co-Büchi games?

Hint: Think of suitably combining Reachability and Safety conditions.

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\text{Buchi}^n(T)$:= "Player 0 can visit T at least n times."

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\text{Buchi}^n(T) :=$ "Player 0 can visit T at least n times."

For $n = 1$: I have to reach T at least once.

Problem

$\text{Buchi}^n(T) :=$ "Player 0 can visit T at least n times."

For $n = 1$: I have to reach T at least once. $\text{Buchi}^1(T) = \text{Reach}(T)$

Problem

$\text{Buchi}^n(T) :=$ "Player 0 can visit T at least n times."

For $n = 1$: I have to reach T at least once. $\text{Buchi}^1(T) = \text{Reach}(T)$

For $n > 1$: I have to reach a vertex in T from which I can force to visit T for $n - 1$ more times.

Problem

$\text{Buchi}^n(T) :=$ "Player 0 can visit T at least n times."

For $n = 1$: I have to reach T at least once. $\text{Buchi}^1(T) = \text{Reach}(T)$

For $n > 1$: I have to reach a vertex in T from which I can force to visit T for $n-1$ more times. $\text{Buchi}^n(T) = \text{Reach}(T \cap \text{force}_0(\text{Buchi}^{n-1}(T)))$

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$\text{Buchi}^n(T) :=$ "Player 0 can visit T at least n times."

For $n = 1$: I have to reach T at least once. $\text{Buchi}^1(T) = \text{Reach}(T)$

For $n > 1$: I have to reach a vertex in T from which I can force to visit T for $n-1$ more times. $\text{Buchi}^n(T) = \text{Reach}(T \cap \text{force}_0(\text{Buchi}^{n-1}(T)))$

Solving Büchi

$\text{Win}_0(\mathcal{G}) = \text{Buchi}(T) :=$ "Player 0 can visit T as much as they wants".

Problem

$\mathrm{Buchi}^n(T)$:= "Player 0 can visit T at least n times."

For $n = 1$: I have to reach T at least once. $\mathrm{Buchi}^1(T) = \mathrm{Reach}(T)$

For $n > 1$: I have to reach a vertex in T from which I can force to visit T for $n - 1$ more times. $\mathrm{Buchi}^n(T) = \mathrm{Reach}(T \cap \mathrm{force}_0(\mathrm{Buchi}^{n-1}(T)))$

Solving Büchi

$\mathrm{Win}_0(\mathcal{G}) = \mathrm{Buchi}(T)$:= "Player 0 can visit T as much as they wants".

$$\mathrm{Win}_0(\mathcal{G}) = \mathrm{Buchi}(T) = \mathrm{Buchi}^1(T) \cap \mathrm{Buchi}^2(T) \cap \ldots \cap \mathrm{Buchi}^n(T) \cap \ldots$$

Problem

$\text{Buchi}^n(T)$:= "Player 0 can visit T at least n times."

For $n = 1$: I have to reach T at least once. $\text{Buchi}^1(T) = \text{Reach}(T)$

For $n > 1$: I have to reach a vertex in T from which I can force to visit T for $n - 1$ more times. $\text{Buchi}^n(T) = \text{Reach}(T \cap \text{force}_0(\text{Buchi}^{n-1}(T)))$

Solving Büchi

$\text{Win}_0(\mathcal{G}) = \text{Buchi}(T)$:= "Player 0 can visit T as much as they wants".

$$\text{Win}_0(\mathcal{G}) = \text{Buchi}(T) = \text{Buchi}^1(T) \cap \text{Buchi}^2(T) \cap \ldots \cap \text{Buchi}^n(T) \cap \ldots$$

$$\text{Buchi}^1(T) \subseteq \text{Buchi}^2(T) \subseteq \ldots \subseteq \text{Buchi}^n(T) \subseteq \ldots$$

SAPIENZA UNIVERSITÀ DI ROMA

Problem

$$\text{Buchi}^n(T) := \text{"Player 0 can visit } T \text{ at least } n \text{ times."}$$

For $n = 1$: I have to reach T at least once. $\text{Buchi}^1(T) = \text{Reach}(T)$

For $n > 1$: I have to reach a vertex in T from which I can force to visit T for $n - 1$ more times. $\text{Buchi}^n(T) = \text{Reach}(T \cap \text{force}_0(\text{Buchi}^{n-1}(T)))$

Solving Büchi

$$\text{Win}_0(\mathcal{G}) = \text{Buchi}(T) := \text{"Player 0 can visit } T \text{ as much as they wants"}.$$

$\text{Win}_0(\mathcal{G}) = \text{Buchi}(T) = \text{Buchi}^1(T) \cap \text{Buchi}^2(T) \cap \ldots \cap \text{Buchi}^n(T) \cap \ldots$

$\text{Buchi}^1(T) \subseteq \text{Buchi}^2(T) \subseteq \ldots \subseteq \text{Buchi}^n(T) \subseteq \ldots$

Fix-point calculation $\nu\mathcal{X}.(\mu\mathcal{Y}.((T \wedge \text{force}_0(\mathcal{X})) \vee \text{force}_0(\mathcal{Y})))$

Consider an arena $\mathbf{A} = (\mathrm{V}, E, \mathrm{V}_0, \mathrm{V}_1)$ and a co-Büchi game $\mathcal{G} = (\mathbf{A}, \mathsf{coBuchi}(T))$.
Define the dual arena $\overline{\mathbf{A}} = (\mathrm{V}, E, \mathrm{V}_1, \mathrm{V}_0)$ and the Büchi game $\overline{\mathcal{G}} = (\overline{\mathbf{A}}, \mathsf{Buchi}(\mathrm{V} \setminus T))$

Exercise - Prove that:

$\mathsf{Win}_0(\mathcal{G}) = \mathsf{Win}_1(\overline{\mathcal{G}})$;

$\mathsf{Win}_1(\mathcal{G}) = \mathsf{Win}_0(\overline{\mathcal{G}})$.

Theorem

We can solve co-Büchi games by solving the dual Büchi game and complement the solution.

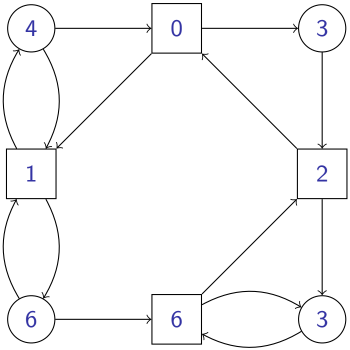
Fix-point calculation $\mu\mathcal{X}.(\nu\mathcal{Y}.((T \vee \mathsf{force}_0(\mathcal{X})) \wedge \mathsf{force}_0(\mathcal{Y})))$

Reachability: $\Diamond T$ $\qquad$ $\mathsf{Reach}(T) = \mu\mathcal{X}.(T \cup \mathsf{force_o}(\mathcal{X}))$

Safety: $\Box T$ $\qquad$ $\mathsf{Safe}(T) = \nu\mathcal{Y}.(T \cap \mathsf{force_o}(\mathcal{Y}))$

Büchi: $\Box\Diamond T$ $\qquad$ $\mathsf{Buchi}(T) = \nu\mathcal{X}.(\mu\mathcal{Y}.((T \wedge \mathsf{force_o}(\mathcal{X})) \vee \mathsf{force_o}(\mathcal{Y})))$

co-Büchi: $\Diamond\Box T$ $\qquad$ $\mathsf{coBuchi}(T) = \mu\mathcal{X}.(\nu\mathcal{Y}.((T \vee \mathsf{force_o}(\mathcal{X})) \wedge \mathsf{force_o}(\mathcal{Y})))$



Every vertex is colored with an natural number. $c : V \to \mathbb{N}$

The play produces an infinite sequence of numbers, aka colors.

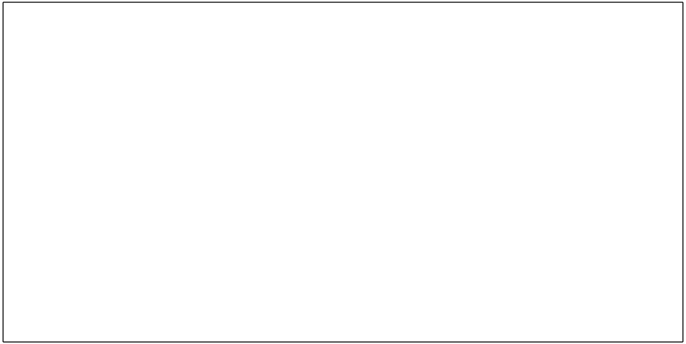
Player 0 wins if the highest color occurring infinitely many times is even.

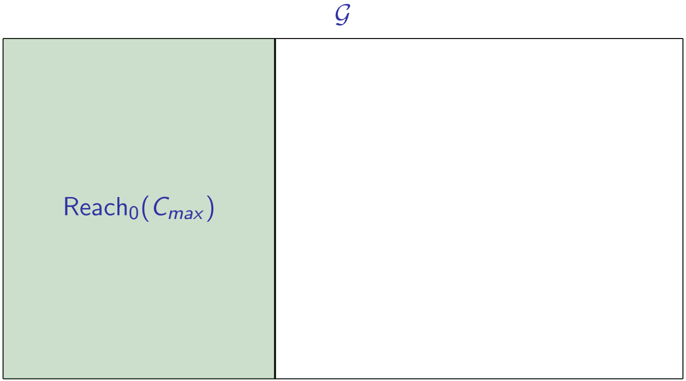
Theorem

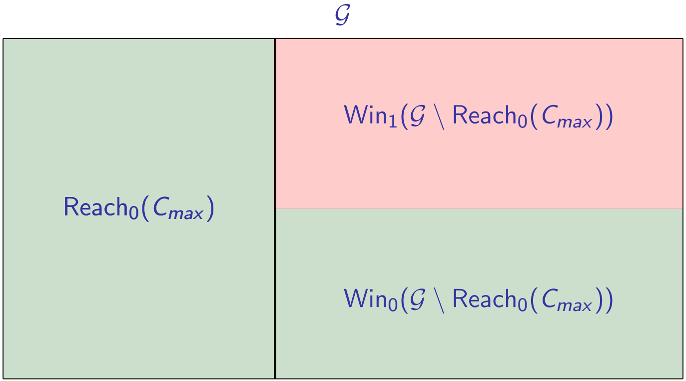
For a given parity game $\mathcal{G}$, computing the winning regions $\mathsf{Win}_0(\mathcal{G})$ and $\mathsf{Win}_1(\mathcal{G})$ can be done in $\mathrm{NP} \cap co\mathrm{NP}$.

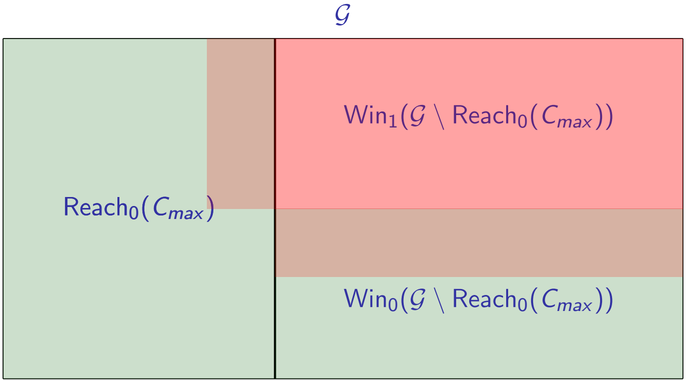
- Determining the right complexity of solving parity games is a long-standing open problem, that has fascinated researchers for more than three decades.
- It has generated a lot of work and it can be considered as a **research topic by itself**!
- The importance of parity games, especially in connection with Synthesis, has spurred the CS community to come up with different approaches for practical efficiency.

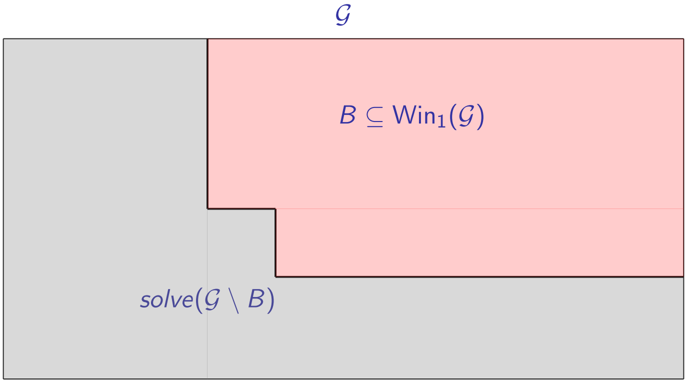
$\mathcal{G}$




$\mathcal{G}$
$\text{Reach}_0(C_{max})$


$\mathcal{G}$
$\text{Win}_1(\mathcal{G} \setminus \text{Reach}_0(C_{max}))$
$\text{Reach}_0(C_{max})$
$\text{Win}_0(\mathcal{G} \setminus \text{Reach}_0(C_{max}))$


$\mathcal{G}$
$\mathsf{Win}_1(\mathcal{G} \setminus \mathsf{Reach}_0(C_{max}))$
$\mathsf{Reach}_0(C_{max})$
$\mathsf{Win}_0(\mathcal{G} \setminus \mathsf{Reach}_0(C_{max}))$


$\mathcal{G}$
$B \subseteq \mathsf{Win}_1(\mathcal{G})$
$solve(\mathcal{G} \setminus B)$

Parity Game (Zielonka's) Algorithm

Algorithm 3 Parity game

p maximal priority in $\mathcal{G}$
if $p = 0$ **then**
    return $\mathsf{Win}_0 = V$; $\mathsf{Win}_1 = \emptyset$
end if
$C_{max} = c^{-1}(p)$ // nodes in $\mathcal{G}$ with highest priority
$i = p \mod 2$ // setting "perspective"
$A = \mathsf{Reach}_i(C_{max})$
$(\mathsf{Win}'_0, \mathsf{Win}'_1) = solve(\mathcal{G} \setminus A)$
if $\mathsf{Win}'_{1-i} = \emptyset$ **then**
    return $\mathsf{Win}_i = V$; $\mathsf{Win}_{1-i} = \emptyset$
end if
$B = \mathsf{Reach}_{1-i}(\mathsf{Win}'_1)$
$(\mathsf{Win}''_0, \mathsf{Win}''_1) = solve(\mathcal{G} \setminus B)$
return $\mathsf{Win}_i = \mathsf{Win}''_i$; $\mathsf{Win}''_{1-i} \cup B$

A standard language for talking about infinite state sequences.

Amir Pnueli - The Temporal Logic of Programs. - FOCS'77

$\top$	truth constant
p	primitive propositions
$\neg\phi$	classical negation
$\phi \vee \psi$	classical disjunction
$\phi \wedge \psi$	classical conjunction

A standard language for talking about infinite state sequences.

Amir Pnueli - The Temporal Logic of Programs. - FOCS'77

$\top$	truth constant	$\bigcirc\phi$	in the next state...
p	primitive propositions	$\Diamond\phi$	will eventually be the case
$\neg\phi$	classical negation	$\Box\phi$	is always the case
$\phi \vee \psi$	classical disjunction	$\phi \mathsf{U} \psi$	ϕ until ψ
$\phi \wedge \psi$	classical conjunction	$\phi \mathsf{R} \psi$	ϕ release ψ

SAPIENZA UNIVERSITÀ DI ROMA

A standard language for talking about infinite state sequences.

Amir Pnueli - The Temporal Logic of Programs. - FOCS'77

$\top$	truth constant	$\bigcirc\phi$	in the next state...
p	primitive propositions	$\Diamond\phi$	will eventually be the case
$\neg\phi$	classical negation	$\Box\phi$	is always the case
$\phi \vee \psi$	classical disjunction	$\phi \mathsf{U} \psi$	ϕ until ψ
$\phi \wedge \psi$	classical conjunction	$\phi \mathsf{R} \psi$	ϕ release ψ

Minimal syntax

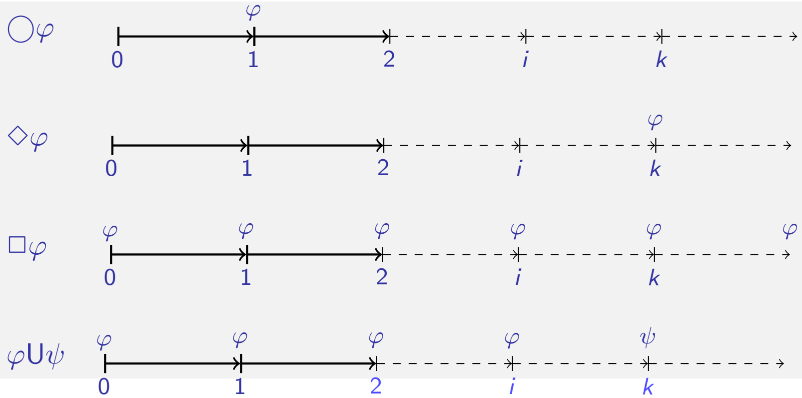
$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi$$

you may encounter the following notations:

$$\begin{aligned} X\varphi &: \bigcirc\varphi \\ F\varphi &: \Diamond\varphi \\ G\varphi &: \Box\varphi \end{aligned}$$

past operators are possible (though not strictly necessary)

Semantics of LTL



LTL formulas are evaluated on infinite traces, that is, obtained from an infinite path.

The language defined by an LTL formula φ is $\mathcal{L}(\varphi) = \{w \in \Sigma^\omega : w \models \varphi\}$.

Example LTL formulae

$\Diamond \mathit{degree}$

eventually I will graduate

Example LTL formulae

$$\Box \neg \mathit{crash}$$

the plane will never crash

Example LTL formulae

$$\Box\Diamond \mathit{eatPizza}$$

$$\Box\Diamond \mathit{eatPizza}$$

I will eat pizza *infinitely often*

SAPIENZA
UNIVERSITÀ DI ROMA

$$\Box\Diamond \mathit{eatPizza}$$

I will eat pizza *infinitely often* (but only in Napoli)

Example LTL formulae

$$\Diamond \Box \mathit{happy}$$

$$\Diamond \Box \mathit{happy}$$

... and they lived happily ever after.

Example LTL formulae

$$(\neg friends) \mathsf{U} youApologise$$

SAPIENZA UNIVERSITÀ DI ROMA

$$(\neg friends) \mathsf{U} youApologise$$

we are not friends *until* you apologise

SAPIENZA UNIVERSITÀ DI ROMA

Describe temporal modalities recursively

- $\varphi \mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc \varphi \mathsf{U} \psi)$ — $\varphi \mathsf{U} \psi$ is a "solution" of $\Psi = \psi \vee (\varphi \wedge \bigcirc \Psi)$
- $\Diamond \psi \equiv \psi \vee \bigcirc \Diamond \psi$ — $\Diamond \psi$ is a solution of $\Psi = \psi \vee \bigcirc \Psi$
- also $\Box \psi \equiv \neg \Diamond \neg \psi \equiv \psi \wedge \bigcirc \Box \psi$ — $\Box \psi$ is a solution of $\Psi = \psi \wedge \bigcirc \Psi$

Define the Release operator R in a way that the following holds:

$\varphi \mathsf{R} \psi \equiv \neg(\neg\varphi \mathsf{U} \neg\psi)$

it also holds that

$\varphi \mathsf{U} \psi \equiv \neg(\neg\varphi \mathsf{R} \neg\psi)$ (Release is dual of Until)

Define the Release operator R in a way that the following holds:

$\varphi \mathsf{R} \psi \equiv \neg(\neg\varphi \mathsf{U} \neg\psi)$

it also holds that

$\varphi \mathsf{U} \psi \equiv \neg(\neg\varphi \mathsf{R} \neg\psi)$ (Release is dual of Until)

PNF

Positive Normal Form for LTL: for $a \in AP$

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{R} \varphi$$

Define the Release operator R in a way that the following holds:

$$\varphi \mathsf{R} \psi \equiv \neg(\neg\varphi \mathsf{U} \neg\psi)$$

it also holds that

$$\varphi \mathsf{U} \psi \equiv \neg(\neg\varphi \mathsf{R} \neg\psi)$$

(Release is dual of Until)

PNF

Positive Normal Form for LTL: for $a \in AP$

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{R} \varphi$$

Theorem

Each LTL formula φ admits an equivalent in PNF sometimes denoted $\mathsf{pnf}(\varphi)$

SAPIENZA UNIVERSITÀ DI ROMA

LTL$_f$

$$\varphi ::= A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \mid \bullet\varphi \mid \Diamond\varphi \mid \Box\varphi \mid Last$$

A: **atomic** propositions

$\neg\varphi$, $\varphi_1 \wedge \varphi_2$: **boolean** connectives

$\bigcirc\varphi$: "**next step exists** and at **next** step (of the trace) φ holds"

$\varphi_1 \mathsf{U} \varphi_2$: "**eventually** φ_2 holds, and φ_1 holds **until** φ_2 does"

$\bullet\varphi \doteq \neg\bigcirc\neg\varphi$: "**if next step exists** then at **next** step φ holds" *(weak next)*

$\Diamond\varphi \doteq \top\mathsf{U}\varphi$: "$\varphi$ will **eventually** hold"

$\Box\varphi \doteq \neg\Diamond\neg\varphi$: "from current till last instant φ will **always** hold"

$Last \doteq \neg\bigcirc\top$: denotes **last** instant of trace.

- $\Diamond \texttt{degree}$
- $\Box \neg \texttt{crash}$
- $\Box \Diamond \texttt{eatPizza}$
- $\Diamond \Box \texttt{happy}$
- $(\neg \texttt{friends}) \mathsf{U} \texttt{youApologise}$

SAPIENZA UNIVERSITÀ DI ROMA

LDL_f

$$\varphi ::= \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \langle\rho\rangle\varphi \mid [\rho]\varphi$$
$$\rho ::= \phi \mid \varphi? \mid \rho_1 + \rho_2 \mid \rho_1; \rho_2 \mid \rho^*$$

ϕ: **propositional formula** on current state/instant

$\neg\varphi$, $\varphi_1 \wedge \varphi_2$: **boolean** connectives

ρ is a **regular expression** on propositional formulas

$\langle\rho\rangle\varphi$: **exists an "execution"** of RE ρ that ends with φ holding

$[\rho]\varphi$: **all "executions"** of RE ρ (along the trace!) end with φ holding

SAPIENZA UNIVERSITÀ DI ROMA

Example

All coffee requests from person p will eventually be served:

$$[\mathtt{true}^*](request_p \supset \langle \mathtt{true}^* \rangle coffee_p)$$

Every time the robot opens door d it closes it immediately after:

$$[\mathtt{true}^*]([openDoor_d]closeDoor_d)$$

Before entering restricted area a the robot must have permission for a:

$$\langle (\neg inArea_a{}^*; getPermission_a; \neg inArea_a{}^*; inArea_a)^*; \neg inArea_a{}^* \rangle end$$

Note that the first two properties (not the third one) can be expressed also in LTL$_f$:

$$\Box(request_p \supset \Diamond coffee_p) \qquad \Box(openDoor_d \supset \bigcirc closeDoor_d)$$