Stackelberg-Pareto Synthesis

Clément Tamines (University of Mons)

Joint work with

Véronique Bruyère (University of Mons)

Jean-François Raskin (Université Libre de Bruxelles)

March 24, 2021

MFV Seminar

Outline of the Talk

Recall **Reactive Synthesis**

- reminder on games
- classical and alternative approaches

Introduce **Stackelberg-Pareto Synthesis**

- our new model
- reactive synthesis in this context

Present **our results**

- fixed-parameter complexity
- NEXPTIME-completeness

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

- given a **specification** for the system

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

- given a **specification** for the system
- synthesize an adequate **controller** for the system

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

- given a **specification** for the system
- synthesize an adequate **controller** for the system
- enforce the specification **whatever the behavior** of the environment

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

- given a **specification** for the system
- synthesize an adequate **controller** for the system
- enforce the specification **whatever the behavior** of the environment

Classical approach for RS [GTW02]

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

- given a **specification** for the system
- synthesize an adequate **controller** for the system
- enforce the specification **whatever the behavior** of the environment

Classical approach for RS [GTW02]

- interaction is modeled using a two-player game

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

- given a **specification** for the system
- synthesize an adequate **controller** for the system
- enforce the specification **whatever the behavior** of the environment

Classical approach for RS [GTW02]

- interaction is modeled using a two-player game
- Player 0 = system, Player 1 = the environment

Reactive Synthesis

Reactive system: system which constantly interacts with its environment

Problem of **Reactive Synthesis** (RS)

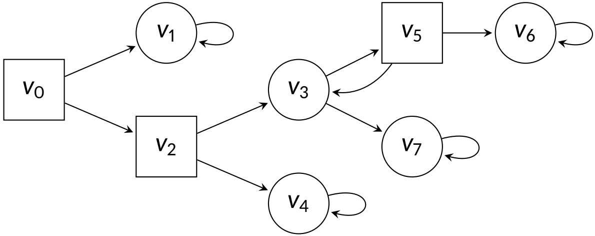
- given a **specification** for the system
- synthesize an adequate **controller** for the system
- enforce the specification **whatever the behavior** of the environment

Classical approach for RS [GTW02]

- interaction is modeled using a two-player game
- Player 0 = system, Player 1 = the environment
- specification = objective

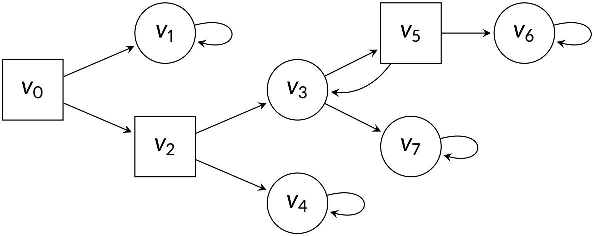
Games: Arenas and Plays

Game Arena: tuple $G = (V, V_0, V_1, E, v_0)$ with (V, E) a directed graph



Games: Arenas and Plays

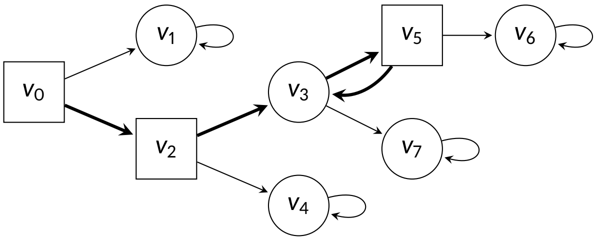
Game Arena: tuple $G = (V, V_0, V_1, E, v_0)$ with (V, E) a directed graph



Play: infinite path starting with the **initial vertex** v_0

Games: Arenas and Plays

Game Arena: tuple $G = (V, V_0, V_1, E, v_0)$ with (V, E) a directed graph

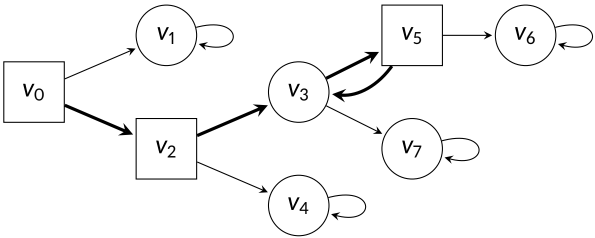


Play: infinite path starting with the **initial vertex** v_0

$\rightarrow \rho = v_0 v_2 (v_3 v_5)^\omega$

Games: Arenas and Plays

Game Arena: tuple $G = (V, V_0, V_1, E, v_0)$ with (V, E) a directed graph



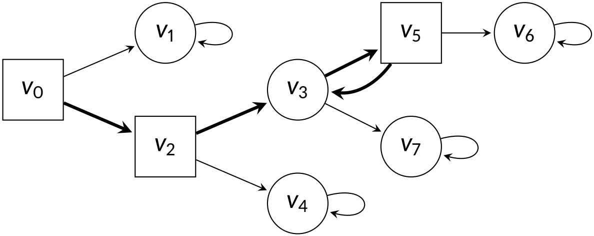
Play: infinite path starting with the **initial vertex** v_0

$\rightarrow \rho = v_0 v_2 (v_3 v_5)^\omega$

History: finite path defined similarly

Games: Arenas and Plays

Game Arena: tuple $G = (V, V_0, V_1, E, v_0)$ with (V, E) a directed graph



Play: infinite path starting with the **initial vertex** v_0

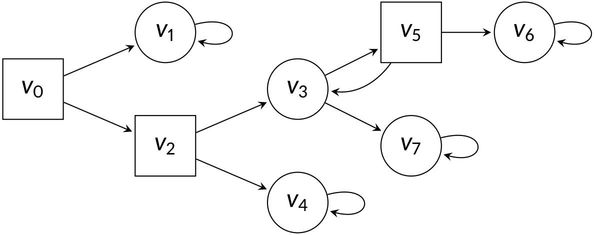
$\rightarrow \rho = v_0 v_2 (v_3 v_5)^\omega$

History: finite path defined similarly

$\rightarrow h = v_0 v_2 v_3 v_5 v_3$

Games: Objectives

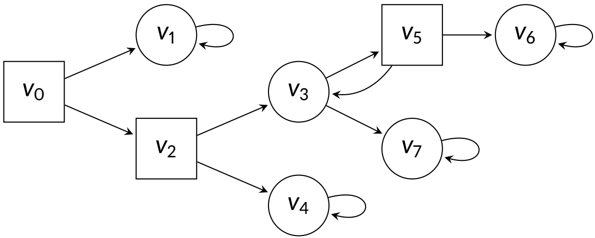
Objective Ω_i for Player $i \in \{0, 1\}$: subset of plays



Games: Objectives

Objective Ω_i for Player $i \in \{0, 1\}$: subset of plays

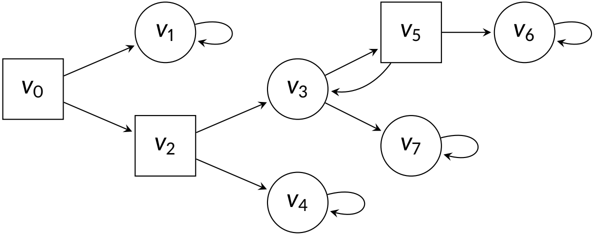
Play ρ **satisfies** objective Ω_i if $\rho \in \Omega_i$



Games: Objectives

Objective Ω_i for Player $i \in \{0, 1\}$: subset of plays

Play ρ **satisfies** objective Ω_i if $\rho \in \Omega_i$

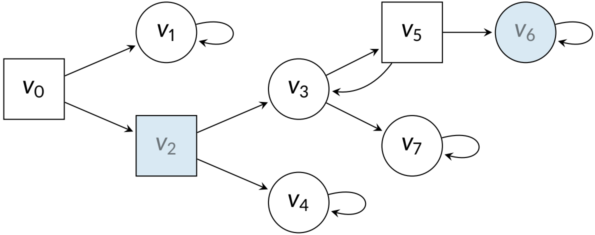


Reachability: plays which *visit* $T \subseteq V$

Games: Objectives

Objective Ω_i for Player $i \in \{0, 1\}$: subset of plays

Play ρ **satisfies** objective Ω_i if $\rho \in \Omega_i$

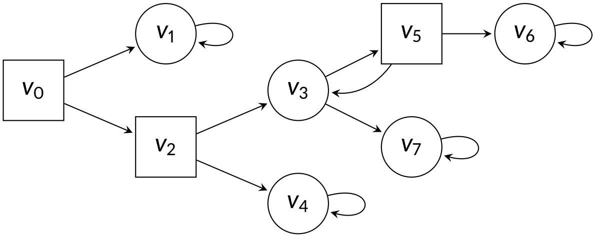


Reachability: plays which *visit* $T \subseteq V$

$\rightarrow T = \{v_2, v_6\}$, $\rho = v_0 v_2 (v_3 v_5)^\omega$ satisfies $\text{Reach}(T)$

Games: Strategies

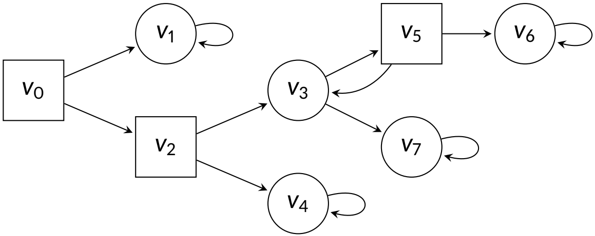
Strategy: $\sigma_i : V^* \times V_i \rightarrow V$ which dictates the choices of Player i



Games: Strategies

Strategy: $\sigma_i : V^* \times V_i \to V$ which dictates the choices of Player i

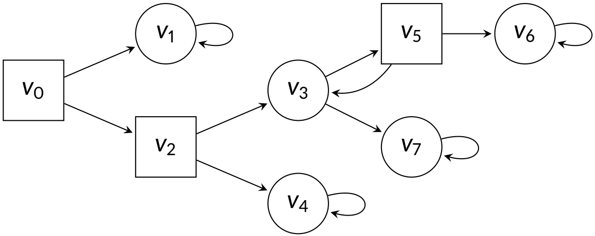
Given $\rho = \underbrace{v_0 v_1 \dots}_{h} \underbrace{v_k}_{\in V_i}$ yields v_{k+1}



Games: Strategies

Strategy: $\sigma_i : V^* \times V_i \rightarrow V$ which dictates the choices of Player i

Given $\rho = \underbrace{v_0 v_1 \dots}_{h} \underbrace{v_k}_{\in V_i}$ yields v_{k+1}

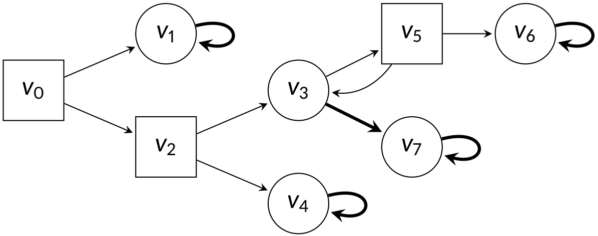


A strategy may use hv_k (**uses memory**) or just v_k (**memoryless**) to yield v_{k+1}

Games: Strategies

Strategy: $\sigma_i : V^* \times V_i \rightarrow V$ which dictates the choices of Player i

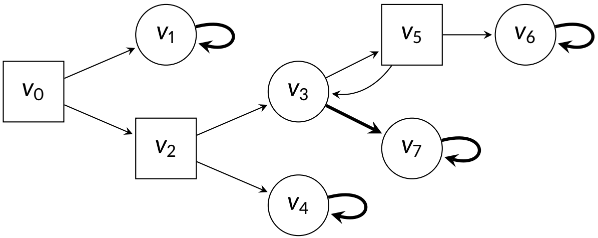
Given $\rho = \underbrace{v_0 v_1 \ldots}_{h} \underbrace{v_k}_{\in V_i}$ yields v_{k+1}



A strategy may use hv_k (**uses memory**) or just v_k (**memoryless**) to yield v_{k+1}

Games: Consistency

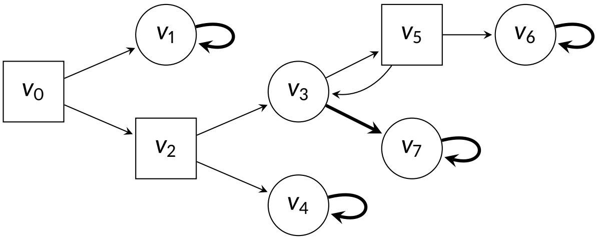
A play is **consistent** with σ_i if $v_{k+1} = \sigma_i(v_0 \ldots v_k)\ \forall k \in \mathbb{N}, \forall v_k \in V_i$



Games: Consistency

A play is **consistent** with σ_i if $v_{k+1} = \sigma_i(v_0 \dots v_k)\ \forall k \in \mathbb{N}, \forall v_k \in V_i$

Consider the **set of plays consistent** with a strategy σ_0

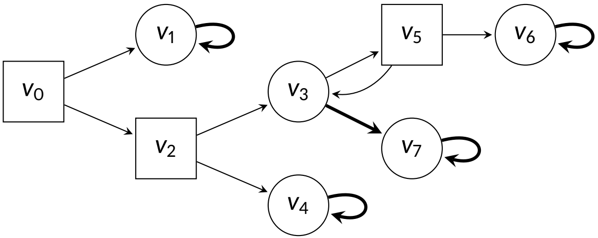


Games: Consistency

A play is **consistent** with σ_i if $v_{k+1} = \sigma_i(v_0 \dots v_k)$ $\forall k \in \mathbb{N}, \forall v_k \in V_i$

Consider the **set of plays consistent** with a strategy σ_0

$\rightarrow \text{Plays}_{\sigma_0} = \{v_0 v_1{}^\omega, v_0 v_2 v_4{}^\omega, v_0 v_2 v_3 v_7{}^\omega\}$

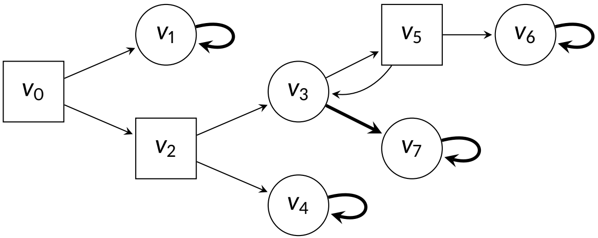


Games: Consistency

A play is **consistent** with σ_i if $v_{k+1} = \sigma_i(v_0 \dots v_k)\ \forall k \in \mathbb{N}, \forall v_k \in V_i$

Consider the **set of plays consistent** with a strategy σ_0

$\rightarrow \text{Plays}_{\sigma_0} = \{v_0 {v_1}^{\omega}, v_0 v_2 {v_4}^{\omega}, v_0 v_2 v_3 {v_7}^{\omega}\}$



A strategy σ_i is **winning for Player *i*** if every play in Plays_{σ_i} satisfies Ω_i

Back to Reactive Synthesis

Classical approach for RS: **zero-sum** games [GTW02]

Back to Reactive Synthesis

Classical approach for RS: **zero-sum** games [GTW02]

- objective of environment is **opposite** objective of system: $\Omega_1 = \neg\Omega_0$

Back to Reactive Synthesis

Classical approach for RS: **zero-sum** games [GTW02]

- objective of environment is **opposite** objective of system: $\Omega_1 = \neg\Omega_0$
- if $\Omega_0 = \text{Reach}(\{v_2, v_6\})$, then $\Omega_1 = \text{Avoid}(\{v_2, v_6\})$

Back to Reactive Synthesis

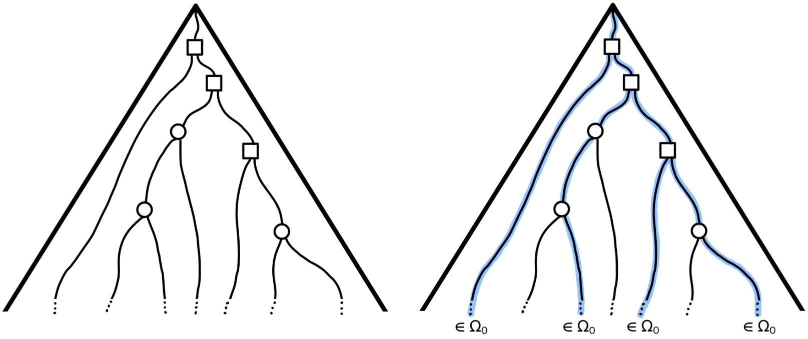
Classical approach for RS: **zero-sum** games [GTW02]

- objective of environment is **opposite** objective of system: $\Omega_1 = \neg\Omega_0$
- if $\Omega_0 = \text{Reach}(\{v_2, v_6\})$, then $\Omega_1 = \text{Avoid}(\{v_2, v_6\})$
- **adversarial environment**: we want a winning strategy for the system

Back to Reactive Synthesis

Classical approach for RS: **zero-sum** games [GTW02]

- objective of environment is **opposite** objective of system: $\Omega_1 = \neg\Omega_0$
- if $\Omega_0 = \text{Reach}(\{v_2, v_6\})$, then $\Omega_1 = \text{Avoid}(\{v_2, v_6\})$
- **adversarial environment**: we want a winning strategy for the system



Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components
- each with **own objective**

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components
- each with **own objective**

Alternative: framework of **Stackelberg games** [vS37] (non-zero-sum)

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components
- each with **own objective**

Alternative: framework of **Stackelberg games** [vS37] (non-zero-sum)

- Player 0 **announces** his strategy σ_0

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components
- each with **own objective**

Alternative: framework of **Stackelberg games** [vS37] (non-zero-sum)

- Player 0 **announces** his strategy σ_0
- Player 1 **rationally answers** with **optimal response** w.r.t. his objective

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components
- each with **own objective**

Alternative: framework of **Stackelberg games** [vS37] (non-zero-sum)

- Player 0 **announces** his strategy σ_0
- Player 1 **rationally answers** with **optimal response** w.r.t. his objective
- goal of Player 0:

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components
- each with **own objective**

Alternative: framework of **Stackelberg games** [vS37] (non-zero-sum)

- Player 0 **announces** his strategy σ_0
- Player 1 **rationally answers** with **optimal response** w.r.t. his objective
- goal of Player 0:
  - announce a strategy that **satisfies his objective**

Setbacks and Alternative

Fully adversarial environment: **bold abstraction of reality**

- assumes the **only goal** of the environment is to **make the system fail**
- environment can be composed of **one or several** components
- each with **own objective**

Alternative: framework of **Stackelberg games** [vS37] (non-zero-sum)

- Player 0 **announces** his strategy σ_0
- Player 1 **rationally answers** with **optimal response** w.r.t. his objective
- goal of Player 0:
  - announce a strategy that **satisfies his objective**
  - **whatever the rational response** of Player 1

Adversarial Rational Synthesis [FKL10, KPV16]

Adversarial Rational Synthesis [FKL10, KPV16]

- **multiplayer** game

Adversarial Rational Synthesis [FKL10, KPV16]

- **multiplayer** game
- Player 0 = system, Players 1 to n = **components** of environment

Adversarial Rational Synthesis [FKL10, KPV16]

- **multiplayer** game
- Player 0 = system, Players 1 to n = **components** of environment
- **rationality**: Players 1 to n settle to a **Nash Equilibrium** (NE), given σ_0

Adversarial Rational Synthesis [FKL10, KPV16]

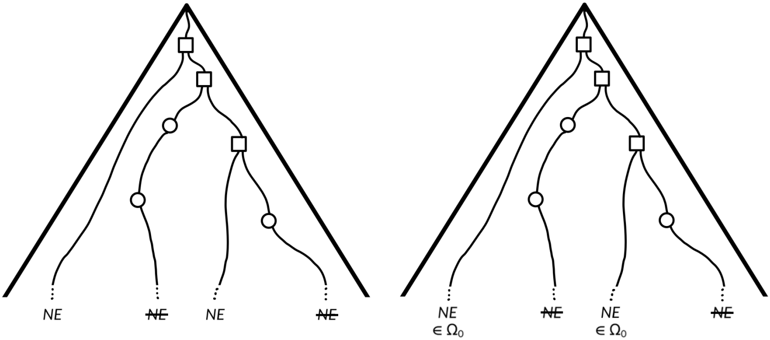
- **multiplayer** game
- Player 0 = system, Players 1 to n = **components** of environment
- **rationality**: Players 1 to n settle to a **Nash Equilibrium** (NE), given σ_0

→ Player 0 **must satisfy** his objective when the environment plays **any** NE

Adversarial Rational Synthesis [FKL10, KPV16]

- **multiplayer** game
- Player 0 = system, Players 1 to n = **components** of environment
- **rationality**: Players 1 to n settle to a **Nash Equilibrium** (NE), given σ_0

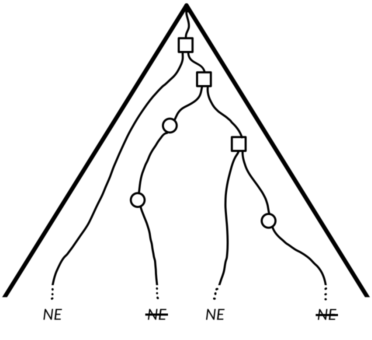
→ Player 0 **must satisfy** his objective when the environment plays **any** NE

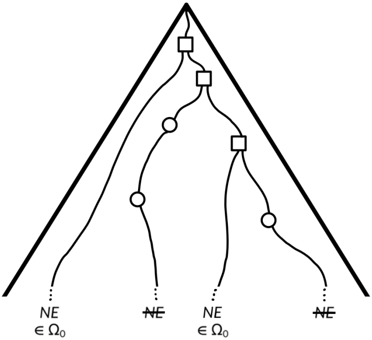


Adversarial Rational Synthesis [FKL10, KPV16]

- **multiplayer** game
- Player 0 = system, Players 1 to n = **components** of environment
- **rationality**: Players 1 to n settle to a **Nash Equilibrium** (NE), given σ_0

⟶ Player 0 **must satisfy** his objective when the environment plays **any** NE





Setbacks: components are independent selfish individuals, no cooperation

Our New Model

Stackelberg-Pareto game (SP game): $\mathcal{G} = (G, \Omega_0, \Omega_1, \dots, \Omega_t)$

Our New Model

Stackelberg-Pareto game (SP game): $\mathcal{G} = (G, \Omega_0, \Omega_1, \ldots, \Omega_t)$

- Player 0 (system): objective Ω_0, announces strategy σ_0

Our New Model

Stackelberg-Pareto game (SP game): $\mathcal{G} = (G, \Omega_0, \Omega_1, \ldots, \Omega_t)$

- Player 0 (system): objective Ω_0, announces strategy σ_0
- Player 1 (environment): **several objectives** $\Omega_1, \ldots, \Omega_t$ (components)

Our New Model

Stackelberg-Pareto game (SP game): $\mathcal{G} = (G, \Omega_0, \Omega_1, \ldots, \Omega_t)$

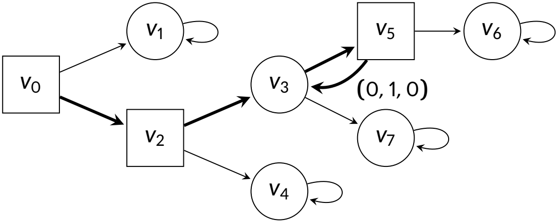
- Player 0 (system): objective Ω_0, announces strategy σ_0
- Player 1 (environment): **several objectives** $\Omega_1, \ldots, \Omega_t$ (components)

Payoff of ρ for Player 1 is the vector of Booleans $\text{pay}(\rho) \in \{0,1\}^t$

$\Omega_1 = \text{Reach}(\{v_6\})$

$\Omega_2 = \text{Reach}(\{v_2\})$

$\Omega_3 = \text{Reach}(\{v_7\})$



Our New Model

Stackelberg-Pareto game (SP game): $\mathcal{G} = (G, \Omega_0, \Omega_1, \dots, \Omega_t)$

- Player 0 (system): objective Ω_0, announces strategy σ_0
- Player 1 (environment): **several objectives** $\Omega_1, \dots, \Omega_t$ (components)

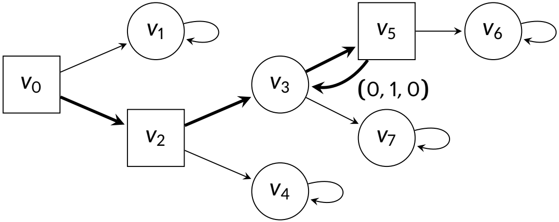
Payoff of ρ for Player 1 is the vector of Booleans $\mathrm{pay}(\rho) \in \{0,1\}^t$

- **order** $\leq$ on payoffs, e.g., $(0,1,0) < (0,1,1)$

$\Omega_1 = \mathrm{Reach}(\{v_6\})$

$\Omega_2 = \mathrm{Reach}(\{v_2\})$

$\Omega_3 = \mathrm{Reach}(\{v_7\})$



Pareto-Optimal Payoffs

1. Player 0 announces his strategy σ_0

Pareto-Optimal Payoffs

1. Player 0 announces his strategy σ_0
2. Player 1 considers Plays_{σ_0}

Pareto-Optimal Payoffs

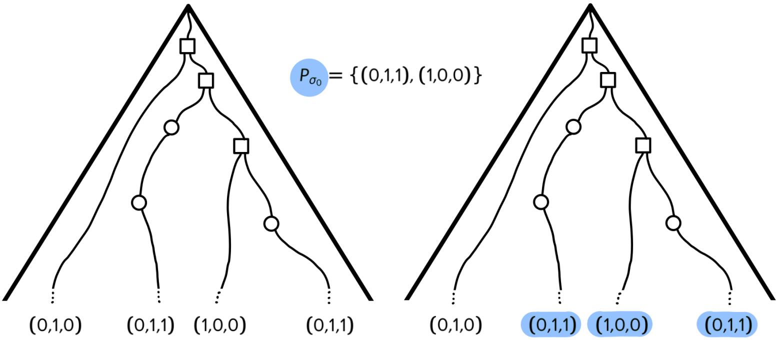
1. Player 0 announces his strategy σ_0
2. Player 1 considers Plays_{σ_0}
    - corresponding **set of payoffs** $\{\text{pay}(\rho) \mid \rho \in \text{Plays}_{\sigma_0}\}$

Pareto-Optimal Payoffs

1. Player 0 announces his strategy σ_0
2. Player 1 considers $\mathsf{Plays}_{\sigma_0}$
   - corresponding **set of payoffs** $\{\mathsf{pay}(\rho) \mid \rho \in \mathsf{Plays}_{\sigma_0}\}$
   - identify **Pareto-optimal** (PO) payoffs (maximal w.r.t. $\leq$) : set P_{σ_0}

Pareto-Optimal Payoffs

1. Player 0 announces his strategy σ_0
2. Player 1 considers Plays_{σ_0}
   - corresponding **set of payoffs** $\{\text{pay}(\rho) \mid \rho \in \text{Plays}_{\sigma_0}\}$
   - identify **Pareto-optimal** (PO) payoffs (maximal w.r.t. $\leq$) : set P_{σ_0}



Stackelberg-Pareto Synthesis Problem

Stackelberg-Pareto Synthesis Problem (SPS problem)

The SPS problem is to decide whether there exists a strategy σ_0 for Player 0 such that for every play $\rho \in \text{Plays}_{\sigma_0}$ with $\text{pay}(\rho) \in P_{\sigma_0}$, it holds that $\rho \in \Omega_0$

Stackelberg-Pareto Synthesis Problem

Stackelberg-Pareto Synthesis Problem (SPS problem)

The SPS problem is to decide whether there exists a strategy σ_0 for Player 0 such that for every play $\rho \in \text{Plays}_{\sigma_0}$ with $\text{pay}(\rho) \in P_{\sigma_0}$, it holds that $\rho \in \Omega_0$

Environment is **rational** and responds to σ_0 **to get a Pareto-optimal payoff**

Stackelberg-Pareto Synthesis Problem

Stackelberg-Pareto Synthesis Problem (SPS problem)

The SPS problem is to decide whether there exists a strategy σ_0 for Player 0 such that for every play $\rho \in \text{Plays}_{\sigma_0}$ with $\text{pay}(\rho) \in P_{\sigma_0}$, it holds that $\rho \in \Omega_0$

Environment is **rational** and responds to σ_0 **to get a Pareto-optimal payoff**
$\longrightarrow$ Player 0 must satisfy Ω_0 in every such rational response

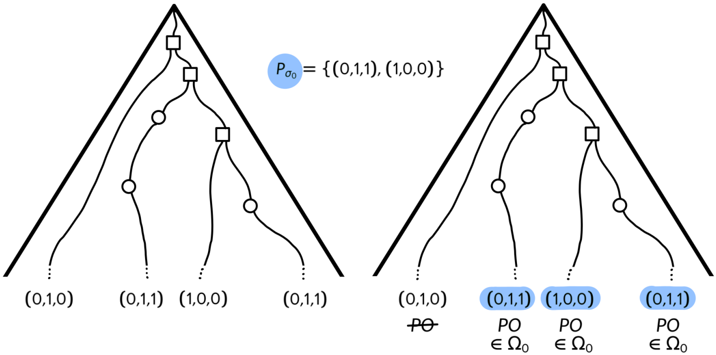
Stackelberg-Pareto Synthesis Problem

Stackelberg-Pareto Synthesis Problem (SPS problem)

The SPS problem is to decide whether there exists a strategy σ_0 for Player 0 such that for every play $\rho \in \mathrm{Plays}_{\sigma_0}$ with $\mathrm{pay}(\rho) \in P_{\sigma_0}$, it holds that $\rho \in \Omega_0$

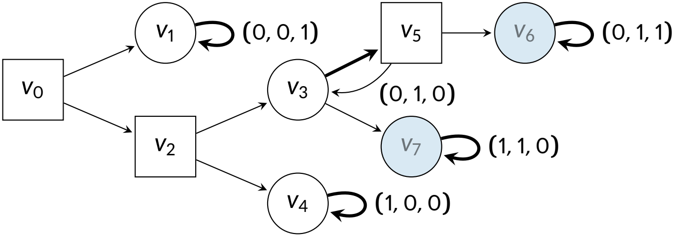
Environment is **rational** and responds to σ_0 **to get a Pareto-optimal payoff**

$\longrightarrow$ Player 0 must satisfy Ω_0 in every such rational response



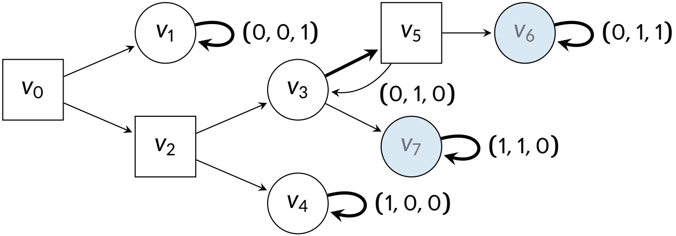
SPS Problem Example (1/2)

Consider σ_0 such that $\sigma_0(v_3) = v_5$



SPS Problem Example (1/2)

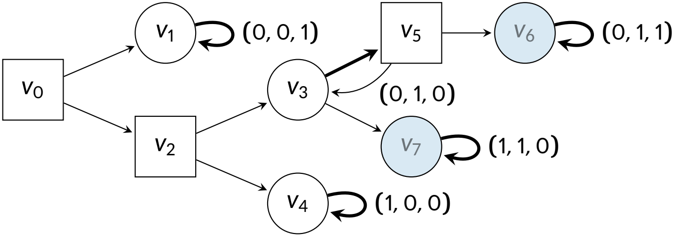
Consider σ_0 such that $\sigma_0(v_3) = v_5$



- $\text{Plays}_{\sigma_0} = \{ \quad v_0 v_1{}^{\omega}, \quad v_0 v_2 v_4{}^{\omega}, \quad v_0 v_2 (v_3 v_5)^{+} v_6{}^{\omega}, \quad v_0 v_2 (v_3 v_5)^{\omega} \ \}$

SPS Problem Example (1/2)

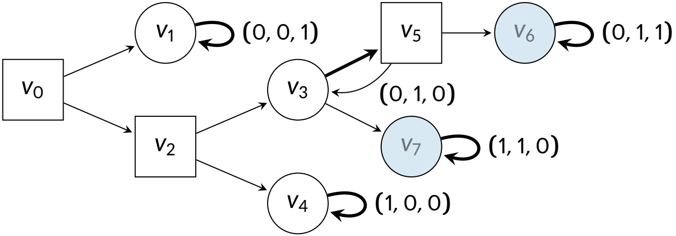
Consider σ_0 such that $\sigma_0(v_3) = v_5$



- $\text{Plays}_{\sigma_0} = \{ \; v_0 v_1{}^{\omega}, \; v_0 v_2 v_4{}^{\omega}, \; v_0 v_2 (v_3 v_5)^+ v_6{}^{\omega}, \; v_0 v_2 (v_3 v_5)^{\omega} \; \}$
- $\text{payoffs} = \{ \, (0,0,1), \; (1,0,0), \; (0,1,1), \; (0,1,0) \, \}$

SPS Problem Example (1/2)

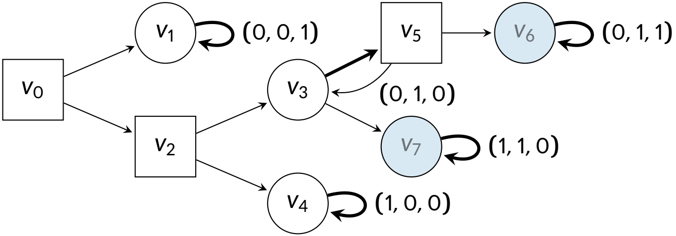
Consider σ_0 such that $\sigma_0(v_3) = v_5$



- $\text{Plays}_{\sigma_0} = \{ \; v_0 v_1{}^\omega, \; v_0 v_2 v_4{}^\omega, \; v_0 v_2 (v_3 v_5)^+ v_6{}^\omega, \; v_0 v_2 (v_3 v_5)^\omega \; \}$
- $\text{payoffs} = \{ \; (0,0,1), \; (1,0,0), \; (0,1,1), \; (0,1,0) \; \}$
- $P_{\sigma_0} = \{ \; (1,0,0), \; (0,1,1) \; \}$

SPS Problem Example (1/2)

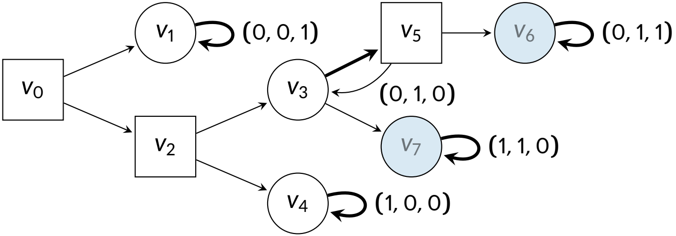
Consider σ_0 such that $\sigma_0(v_3) = v_5$



- $\mathsf{Plays}_{\sigma_0} = \{ \ v_0v_1{}^\omega, \ v_0v_2v_4{}^\omega, \ v_0v_2(v_3v_5)^+v_6{}^\omega, \ v_0v_2(v_3v_5)^\omega \ \}$
- $\mathsf{payoffs} = \{ \ (0,0,1), \ (1,0,0), \ (0,1,1), \ (0,1,0) \ \}$
- $P_{\sigma_0} = \{ \ (1,0,0), \ (0,1,1) \ \}$
- σ_0 is **not a solution** to the SPS problem, e.g., $\rho = v_0v_2(v_4)^\omega \notin \Omega_0$

SPS Problem Example (1/2)

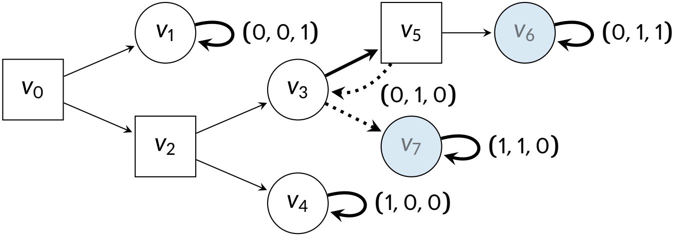
Consider σ_0 such that $\sigma_0(v_3) = v_5$



- $\text{Plays}_{\sigma_0} = \{ \; v_0v_1{}^\omega, \; v_0v_2v_4{}^\omega, \; v_0v_2(v_3v_5)^+v_6{}^\omega, \; v_0v_2(v_3v_5)^\omega \; \}$
- $\text{payoffs} = \{ \; (0,0,1), \; (1,0,0), \; (0,1,1), \; (0,1,0) \; \}$
- $P_{\sigma_0} = \{ \; (1,0,0), \; (0,1,1) \; \}$
- σ_0 is **not a solution** to the SPS problem, e.g., $\rho = v_0v_2(v_4)^\omega \notin \Omega_0$
  $\longrightarrow$ the only other **memoryless** strategy is **not a solution either**

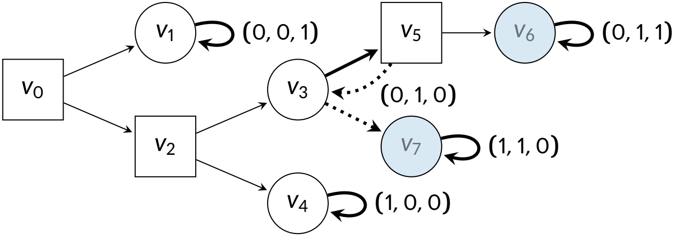
SPS Problem Example (2/2)

Finite-memory strategy σ_0' s.t. $\sigma_0'(v_0v_2v_3) = v_5$ and $\sigma_0'(v_0v_2v_3v_5v_3) = v_7$



SPS Problem Example (2/2)

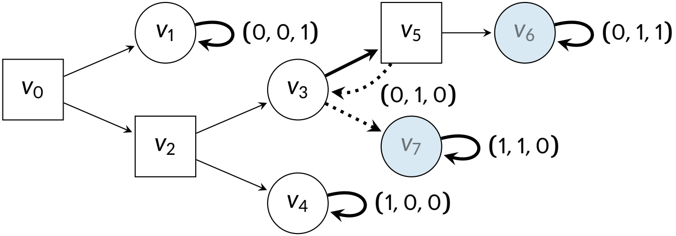
Finite-memory strategy σ'_0 s.t. $\sigma'_0(v_0v_2v_3) = v_5$ and $\sigma'_0(v_0v_2v_3v_5v_3) = v_7$



σ'_0 **is a solution** to the SPS problem: $\rho \in \Omega_0$ when $\text{pay}(\rho) \in P_{\sigma'_0}$

SPS Problem Example (2/2)

Finite-memory strategy σ_0' s.t. $\sigma_0'(v_0 v_2 v_3) = v_5$ and $\sigma_0'(v_0 v_2 v_3 v_5 v_3) = v_7$



σ_0' **is a solution** to the SPS problem: $\rho \in \Omega_0$ when $\text{pay}(\rho) \in P_{\sigma_0'}$

$\longrightarrow$ Player 0 **may need memory** to have a solution to the SPS problem

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

- introduce the model and problem

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

- introduce the model and problem
- consider SP games where

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

- introduce the model and problem
- consider SP games where
  - every objective is parity: **parity SP games**

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

- introduce the model and problem
- consider SP games where
  - every objective is parity: **parity SP games**
    $\longrightarrow$ can model general class of ω-regular objectives

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

- introduce the model and problem
- consider SP games where
  - every objective is parity: **parity SP games**
    $\longrightarrow$ can model general class of ω-regular objectives
  - every objective is reachability: **reachability SP games**

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

- introduce the model and problem
- consider SP games where
  - every objective is parity: **parity SP games**
    → can model general class of ω-regular objectives
  - every objective is reachability: **reachability SP games**
    → simpler setting

Our results

Stackelberg-Pareto Synthesis (submitted to ICALP 2021) [BRT21]

- introduce the model and problem
- consider SP games where
  - every objective is parity: **parity SP games**
    → can model general class of ω-regular objectives
  - every objective is reachability: **reachability SP games**
    → simpler setting
- thorough analysis of the complexity of solving the SPS problem

Fixed-Parameter Complexity [DF12]

A problem is **fixed-parameter tractable** (FPT) for parameter k if there exists a solution running in $f(k) \times n^{\mathcal{O}(1)}$ where f is a function of k independent of n

Fixed-Parameter Complexity [DF12]

A problem is **fixed-parameter tractable** (FPT) for parameter k if there exists a solution running in $f(k) \times n^{\mathcal{O}(1)}$ where f is a function of k independent of n

Example: solving a problem is polynomial in input size, **exponential in k**

Fixed-Parameter Complexity [DF12]

A problem is **fixed-parameter tractable** (FPT) for parameter k if there exists a solution running in $f(k) \times n^{\mathcal{O}(1)}$ where f is a function of k independent of n

Example: solving a problem is polynomial in input size, **exponential in k**
$\longrightarrow$ solving the problem is fixed-parameter tractable (easy if fix a small k)

Fixed-Parameter Complexity [DF12]

A problem is **fixed-parameter tractable** (FPT) for parameter k if there exists a solution running in $f(k) \times n^{\mathcal{O}(1)}$ where f is a function of k independent of n

Example: solving a problem is polynomial in input size, **exponential in k**
$\longrightarrow$ solving the problem is fixed-parameter tractable (easy if fix a small k)

Fixed-Parameter Complexity of SP games

Solving the SPS problem is FPT for reachability SP games for parameter t (number of objectives of Player 1) and FPT for parity SP games for parameters t and the maximal priority according to each parity objective of Player 1

Fixed-Parameter Complexity [DF12]

A problem is **fixed-parameter tractable** (FPT) for parameter k if there exists a solution running in $f(k) \times n^{\mathcal{O}(1)}$ where f is a function of k independent of n

Example: solving a problem is polynomial in input size, **exponential in k**
$\longrightarrow$ solving the problem is fixed-parameter tractable (easy if fix a small k)

Fixed-Parameter Complexity of SP games

Solving the SPS problem is FPT for reachability SP games for parameter t (number of objectives of Player 1) and FPT for parity SP games for parameters t and the maximal priority according to each parity objective of Player 1

Sound: in practice, we can assume those parameters to have **small values**

Challenger-Prover Game

To show FPT results: reduction to **Challenger-Prover game** (C-P game)

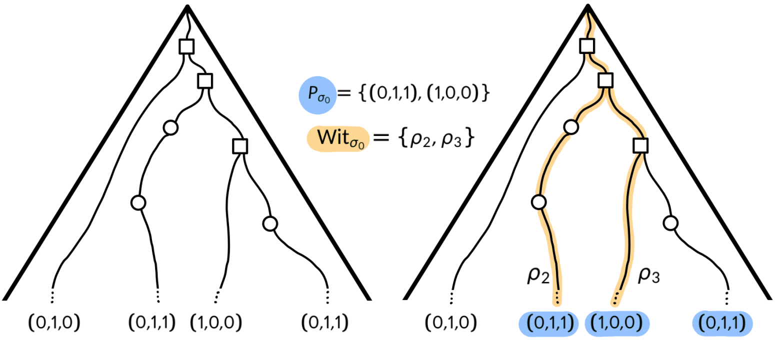
- **two-player zero-sum game** $\mathcal{G}'$, created from $\mathcal{G}$
- played between **Challenger** ($\mathcal{C}$) and **Prover** ($\mathcal{P}$)
- **solution** to the SPS problem in $\mathcal{G} \iff$ **winning strategy** for $\mathcal{P}$ in $\mathcal{G}'$
- described in a **generic way**, later **adapted** to parity/reachability

Intuition: $\mathcal{P}$ tries to **show** the existence of a solution, $\mathcal{C}$ tries to **disprove** it

Witnesses

C-P game uses important notion of **witness**

- given σ_0, we have the set P_{σ_0} of **PO payoffs**
- for each $p \in P_{\sigma_0}$, **there exists** ρ s.t. $\mathrm{pay}(\rho) = p$
- select **one such** ρ for each $p \in P_{\sigma_0}$ (**witness of** p): set Wit_{σ_0}

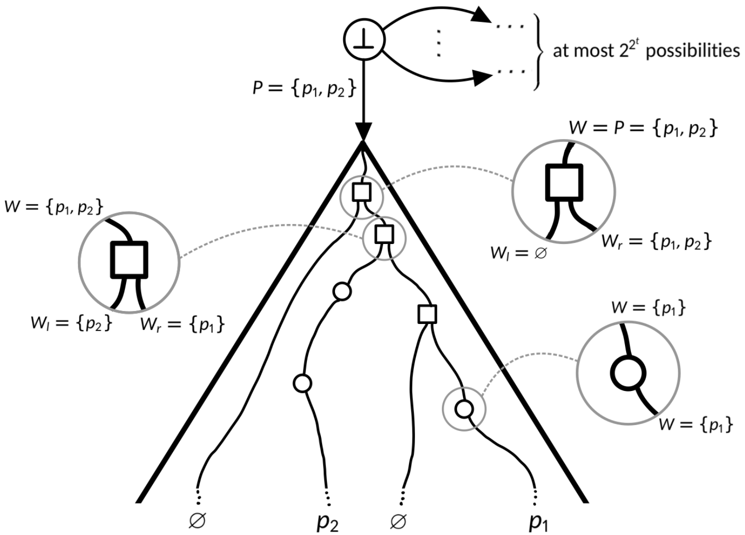


Intuition on the C-P game

w.l.o.g. we consider SP games s.t. each vertex has at most **two successors**

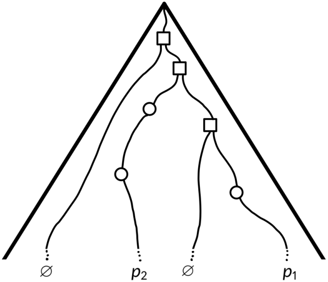
1. $\mathcal{P}$ **selects a set** P of payoffs, he announces it is P_{σ_0} for σ_0 he is building
2. $\mathcal{P}$ tries to show the existence of a **set of witnesses for** P
3. After selection, **one-to-one correspondence** between plays in $\mathcal{G}$ and $\mathcal{G}'$
   - vertices in $\mathcal{G}'$ are **augmented with a set** W which is a subset of P
   - initially $W = P$
   - after history in $\mathcal{G}'$, W **contains** p if the **corresponding history** in $\mathcal{G}$ is **prefix of the witness for** p in the set Wit_{σ_0} that $\mathcal{P}$ is building

Witnesses in the C-P Game



Objective in the C-P Game

Given a play ρ' in $\mathcal{G}'$, there is a corresponding play ρ in $\mathcal{G}$



If play ρ guessed to have payoff p **(1)**

- check that $\text{pay}(\rho) = p$
- check that $\rho \in \Omega_0$

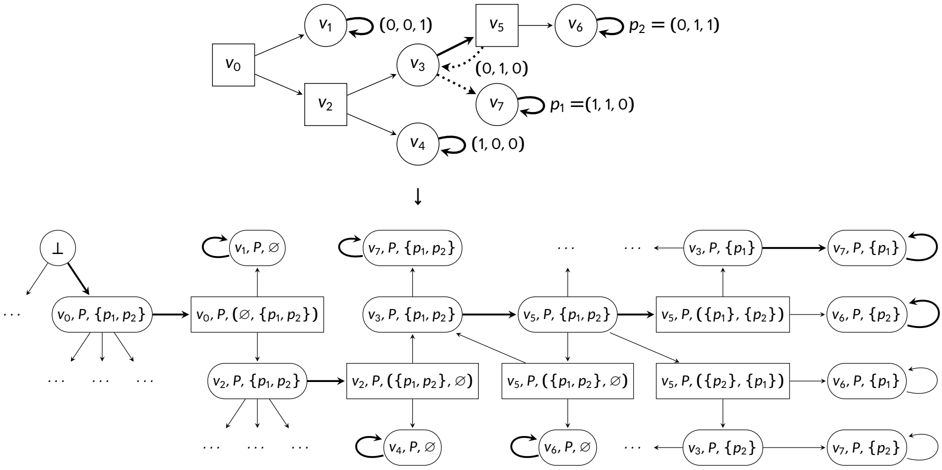
Otherwise

- if $\text{pay}(\rho) = p \in P$, check that $\rho \in \Omega_0$ **(2)**
- else check $\text{pay}(\rho) < p \in P$ **(3)**

Reachability SP game: augment the arena with **set of satisfied objectives** $\longrightarrow$ checking (1-3) = **Büchi objective**

Parity SP game: checking (1-3) = **Boolean combination of Büchi objectives**

C-P Game for our Running Example



Complexity Class

NEXPTIME-Membership

The SPS problem is in NEXPTIME for reachability and parity SP games

Complexity Class

NEXPTIME-Membership

The SPS problem is in NEXPTIME for reachability and parity SP games

Intuition: use **important result on the strategies** which are solution

Complexity Class

NEXPTIME-Membership

The SPS problem is in NEXPTIME for reachability and parity SP games

Intuition: use **important result on the strategies** which are solution

- if Player 0 has a solution, he has a **finite-memory one**

Complexity Class

NEXPTIME-Membership

The SPS problem is in NEXPTIME for reachability and parity SP games

Intuition: use **important result on the strategies** which are solution

- if Player 0 has a solution, he has a **finite-memory one**
- with at most an **exponential number** of memory states

Complexity Class

NEXPTIME-Membership

The SPS problem is in NEXPTIME for reachability and parity SP games

Intuition: use **important result on the strategies** which are solution

- if Player 0 has a solution, he has a **finite-memory one**
- with at most an **exponential number** of memory states

Membership: NEXPTIME **algorithm** where

Complexity Class

NEXPTIME-Membership

The SPS problem is in NEXPTIME for reachability and parity SP games

Intuition: use **important result on the strategies** which are solution

- if Player 0 has a solution, he has a **finite-memory one**
- with at most an **exponential number** of memory states

Membership: NEXPTIME **algorithm** where

- non-deterministically **guess** a strategy (with exponential size)

Complexity Class

NEXPTIME-Membership

The SPS problem is in NEXPTIME for reachability and parity SP games

Intuition: use **important result on the strategies** which are solution

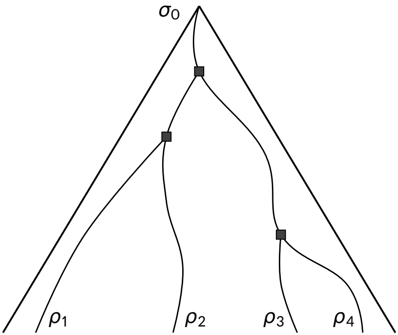
- if Player 0 has a solution, he has a **finite-memory one**
- with at most an **exponential number** of memory states

Membership: NEXPTIME **algorithm** where

- non-deterministically **guess** a strategy (with exponential size)
- **check** that it is a solution **in exponential time** (using automaton)

Constructing a Finite-Memory Strategy: σ_0

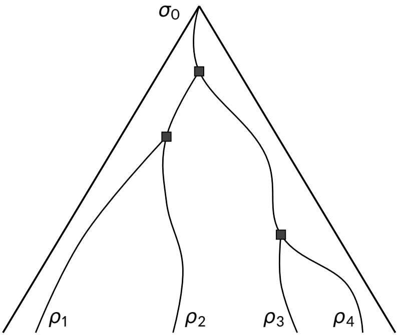
Start from a solution σ_0 to the SPS problem with $\mathrm{Wit}_{\sigma_0} = \{\rho_1, \rho_2, \rho_3, \rho_4\}$



Constructing a Finite-Memory Strategy: σ_0

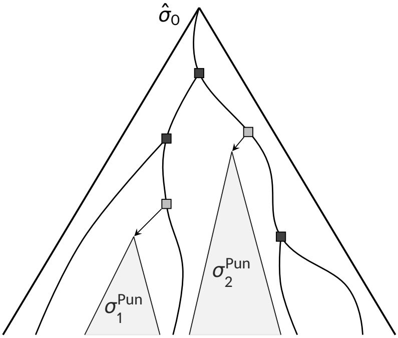
Start from a solution σ_0 to the SPS problem with $\mathrm{Wit}_{\sigma_0} = \{\rho_1, \rho_2, \rho_3, \rho_4\}$

Intuition: build exponential-size strategy which yields $\{c\rho_1, c\rho_2, c\rho_3, c\rho_4\}$



Constructing a Finite-Memory Strategy: $\hat{\sigma}_0$

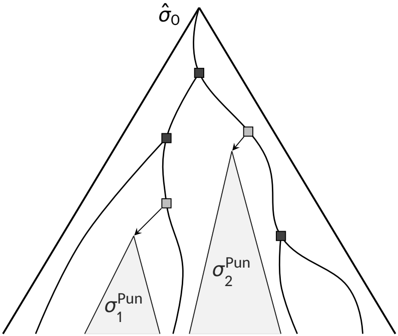
$\hat{\sigma}_0$ follows σ_0 in prefix of witness, on deviation switch to **punishing strategy**



Constructing a Finite-Memory Strategy: $\hat{\sigma}_0$

$\hat{\sigma}_0$ follows σ_0 in prefix of witness, on deviation switch to **punishing strategy**

σ^{Pun} imposes Ω_0 or $\cancel{PO}$, this makes the deviation **irrational** for Player 1

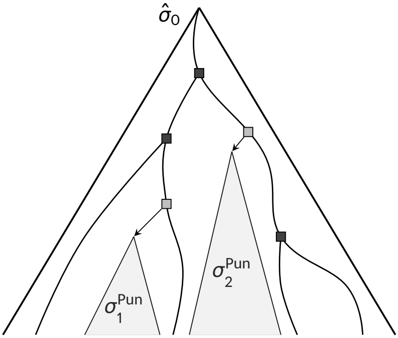


Constructing a Finite-Memory Strategy: $\hat{\sigma}_0$

$\hat{\sigma}_0$ follows σ_0 in prefix of witness, on deviation switch to **punishing strategy**

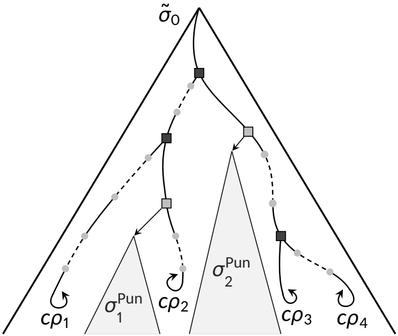
σ^{Pun} imposes Ω_0 or $\cancel{PO}$, this makes the deviation **irrational** for Player 1

Exponentially many different punishing strategies, with **exponential size**



Constructing a Finite-Memory Strategy: $\tilde{\sigma}_0$

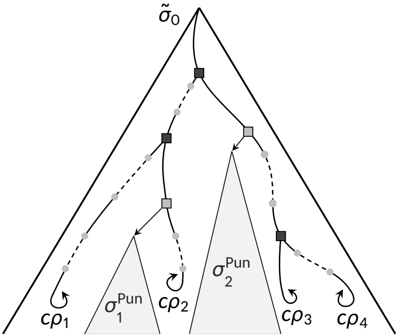
Decompose each witness in Wit_{σ_0} into at most **exponentially many** parts



Constructing a Finite-Memory Strategy: $\tilde{\sigma}_0$

Decompose each witness in Wit_{σ_0} into at most **exponentially many** parts

Compact parts into finite elementary paths or lassos of polynomial length



The SPS problem is NP-hard on Tree Arenas

Simple setting of **tree arenas**: trees with loops on leaves

The SPS problem is NP-hard on Tree Arenas

Simple setting of **tree arenas**: trees with loops on leaves

NP-hardness is shown using the **Set Cover problem** (NP-complete) [Kar72]

The SPS problem is NP-hard on Tree Arenas

Simple setting of **tree arenas**: trees with loops on leaves

NP-hardness is shown using the **Set Cover problem** (NP-complete) [Kar72]

- $C = \{e_1, e_2, \ldots, e_n\}$ of n **elements**

The SPS problem is NP-hard on Tree Arenas

Simple setting of **tree arenas**: trees with loops on leaves

NP-hardness is shown using the **Set Cover problem** (NP-complete) [Kar72]

- $C = \{e_1, e_2, \ldots, e_n\}$ of n **elements**
- m **subsets** $S_1, S_2, \ldots, S_m$ s.t. $S_i \subseteq C$

The SPS problem is NP-hard on Tree Arenas

Simple setting of **tree arenas**: trees with loops on leaves

NP-hardness is shown using the **Set Cover problem** (NP-complete) [Kar72]

- $C = \{e_1, e_2, \ldots, e_n\}$ of n **elements**
- m **subsets** $S_1, S_2, \ldots, S_m$ s.t. $S_i \subseteq C$
- an **integer k** $\leq m$

The SPS problem is NP-hard on Tree Arenas

Simple setting of **tree arenas**: trees with loops on leaves

NP-hardness is shown using the **Set Cover problem** (NP-complete) [Kar72]

- $C = \{e_1, e_2, \ldots, e_n\}$ of n **elements**
- m **subsets** $S_1, S_2, \ldots, S_m$ s.t. $S_i \subseteq C$
- an **integer k** $\leq m$
- **find** k **indexes** $i_1, i_2, \ldots, i_k$ s.t. $C = \bigcup_{j=1}^{k} S_{i_j}$.

The SPS problem is NP-hard on Tree Arenas

Simple setting of **tree arenas**: trees with loops on leaves

NP-hardness is shown using the **Set Cover problem** (NP-complete) [Kar72]

- $C = \{e_1, e_2, \dots, e_n\}$ of n **elements**
- m **subsets** $S_1, S_2, \dots, S_m$ s.t. $S_i \subseteq C$
- an **integer k** $\leq m$
- **find** k **indexes** $i_1, i_2, \dots, i_k$ s.t. $C = \bigcup_{j=1}^{k} S_{i_j}$.

Use an **SP game with polynomial number of vertices** such that there is a solution to the SC problem $\iff$ Player 0 has a solution to the SPS problem

Reduction to the Set Cover Problem

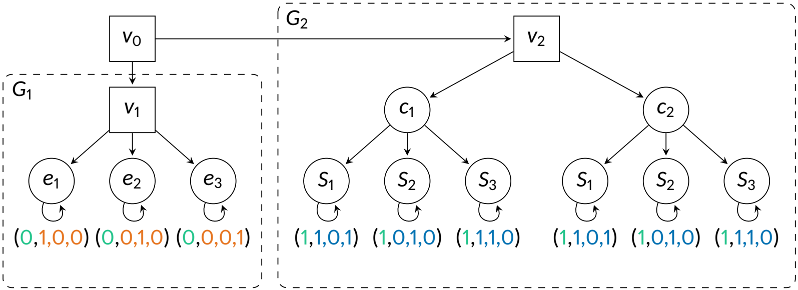
Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$

Reduction to the Set Cover Problem

Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$
special objective (Ω_1) and one objective per element ($\Omega_2, \Omega_3, \Omega_4$)

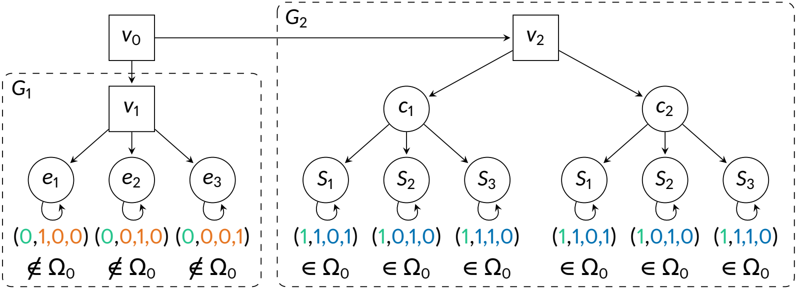
Reduction to the Set Cover Problem

Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$
special objective (Ω_1) and one objective per element ($\Omega_2, \Omega_3, \Omega_4$)



Reduction to the Set Cover Problem

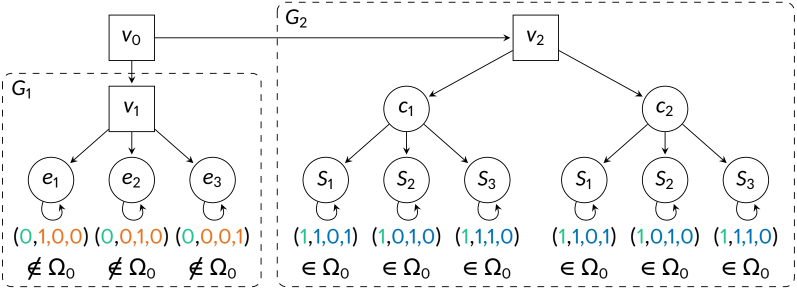
Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$
special objective (Ω_1) and one objective per element ($\Omega_2, \Omega_3, \Omega_4$)



Every play in G_1 is **consistent with any strategy** of Player 0 and $\notin \Omega_0$

Reduction to the Set Cover Problem

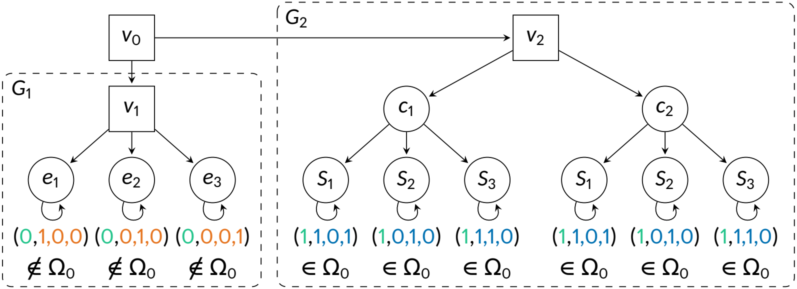
Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$
special objective (Ω_1) and one objective per element ($\Omega_2, \Omega_3, \Omega_4$)



Every play in G_1 is **consistent with any strategy** of Player 0 and $\notin \Omega_0$
$\longrightarrow$ in a solution, payoffs from G_1 **cannot be PO**

Reduction to the Set Cover Problem

Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$
special objective (Ω_1) and one objective per element ($\Omega_2, \Omega_3, \Omega_4$)

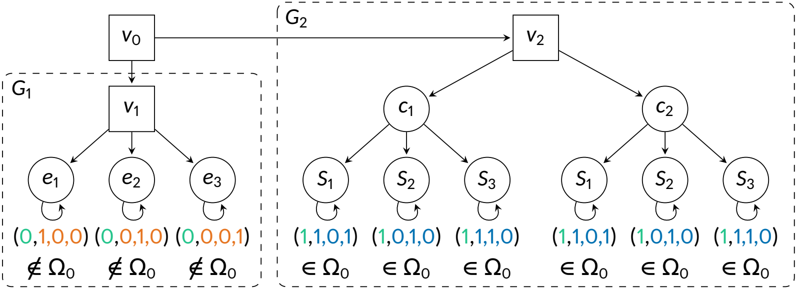


Every play in G_1 is **consistent with any strategy** of Player 0 and $\notin \Omega_0$
$\longrightarrow$ in a solution, payoffs from G_1 **cannot be PO**

Each payoff in G_1 must be **< than some payoff in G_2** (corresponding to a set)

Reduction to the Set Cover Problem

Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$
special objective (Ω_1) and one objective per element ($\Omega_2, \Omega_3, \Omega_4$)

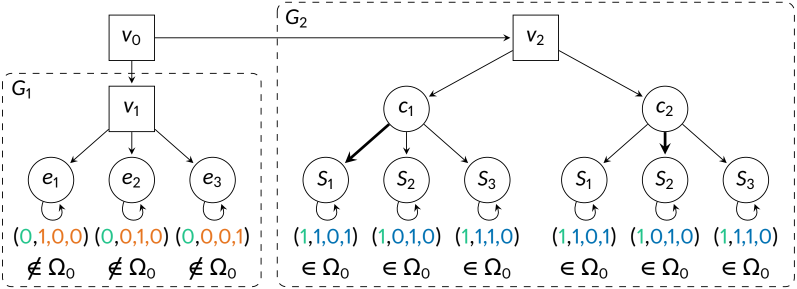


Every play in G_1 is **consistent with any strategy** of Player 0 and $\notin \Omega_0$
$\longrightarrow$ in a solution, payoffs from G_1 **cannot be PO**

Each payoff in G_1 must be **$<$ than some payoff in G_2** (corresponding to a set)

Reduction to the Set Cover Problem

Example: $C = \{e_1, e_2, e_3\}$, $S_1 = \{e_1, e_3\}$, $S_2 = \{e_2\}$, $S_3 = \{e_1, e_2\}$, $k = 2$
special objective (Ω_1) and one objective per element ($\Omega_2, \Omega_3, \Omega_4$)



Every play in G_1 is **consistent with any strategy** of Player 0 and $\notin \Omega_0$
$\longrightarrow$ in a solution, payoffs from G_1 **cannot be PO**

Each payoff in G_1 must be **$<$ than some payoff in G_2** (corresponding to a set)

Hardness

NEXPTIME-Hardness

The SPS problem is NEXPTIME-hard for reachability and parity SP games

Hardness

NEXPTIME-Hardness

The SPS problem is NEXPTIME-hard for reachability and parity SP games

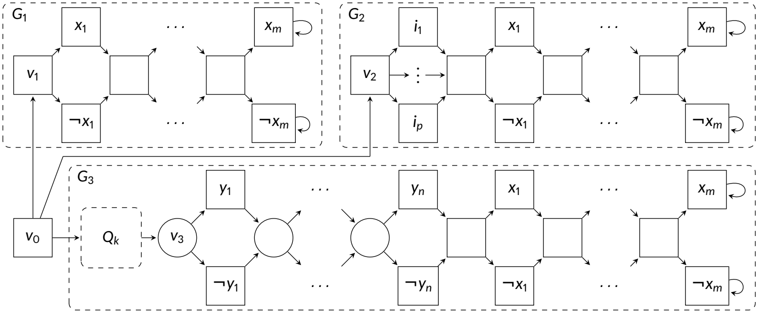
Intuition: use **succinct variant** of Set Cover problem (NEXPTIME-complete)
$\longrightarrow$ Set Cover problem succinctly defined using **CNF formulas**

Hardness

NEXPTIME-Hardness

The SPS problem is NEXPTIME-hard for reachability and parity SP games

Intuition: use **succinct variant** of Set Cover problem (NEXPTIME-complete)
⟶ Set Cover problem succinctly defined using **CNF formulas**



Conclusion

Recalled the concept of **reactive synthesis**

- classical approach using two-player zero-sum games
- setbacks and alternatives

Introduced **Stackelberg-Pareto Synthesis problem**

- our novel approach to synthesis
- FPT and NEXPTIME-completeness

Future work

- study other ω-regular objectives
- adapt to quantitative objectives such as mean-payoff
- study whether rational synthesis can benefit from our approaches

Bibliography I

[BRT21] Véronique Bruyère, Jean-François Raskin, and Clément Tamines.
Stackelberg-pareto synthesis (full version).
CoRR, abs/2102.08925, 2021.

[DF12] R.G. Downey and M.R. Fellows.
Parameterized Complexity.
Monographs in Computer Science. Springer New York, 2012.

[FKL10] Dana Fisman, Orna Kupferman, and Yoad Lustig.
Rational synthesis.
In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS*

Bibliography II

2010, Paphos, Cyprus, March 20-28, 2010. Proceedings, volume 6015 of *Lecture Notes in Computer Science*, pages 190–204. Springer, 2010.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors.
Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[Kar72] Richard M. Karp.
Reducibility among combinatorial problems.
In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J.*

Bibliography III

Watson Research Center, Yorktown Heights, New York, USA, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[KPV16] Orna Kupferman, Giuseppe Perelli, and Moshe Y. Vardi.
Synthesis with rational environments.
Ann. Math. Artif. Intell., 78(1):3–20, 2016.

[vS37] Heinrich Freiherr von Stackelberg.
Marktform und Gleichgewicht.
Wien und Berlin, J. Springer, 1937.