

Stopping Criterion for Strategy Improvement on Concurrent Stochastic Games

Eline Bovy¹, Marta Grobelna², Jan Křetínský^{3,2}, Maximilian Weininger^{4,2}

1) Radboud University 2) Technical University of Munich 3) Masaryk University 4) Ruhr-University Bochum

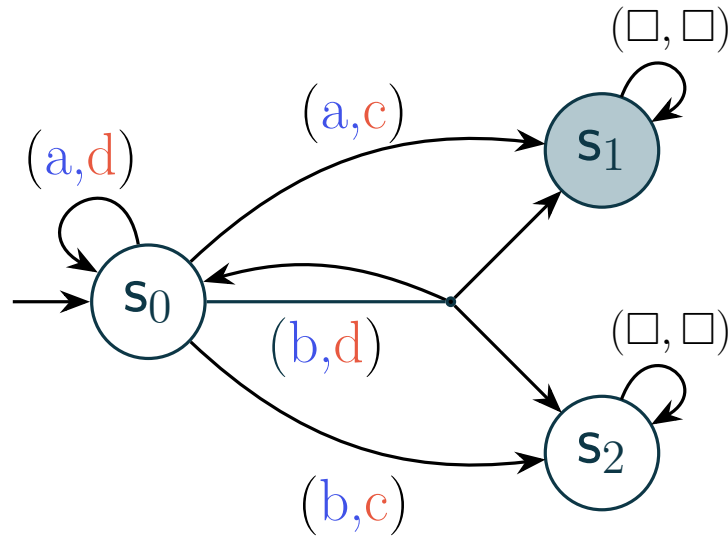
Highlights 2026

Vienna

Radboud Universiteit

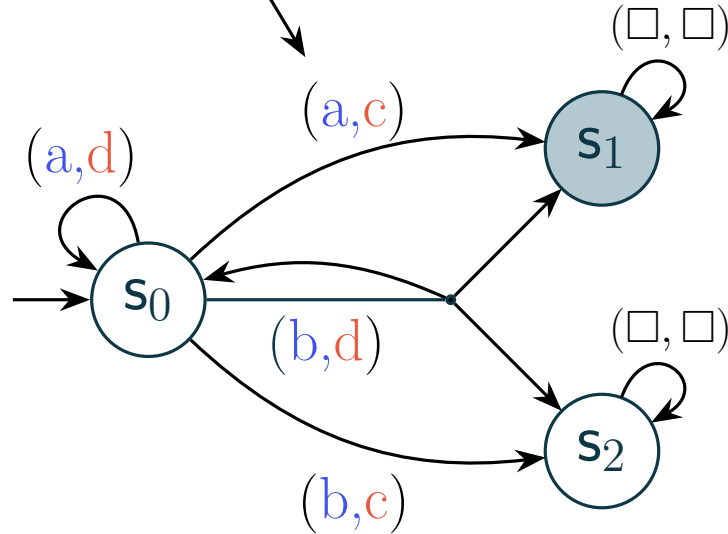


Concurrent Stochastic Reachability Games

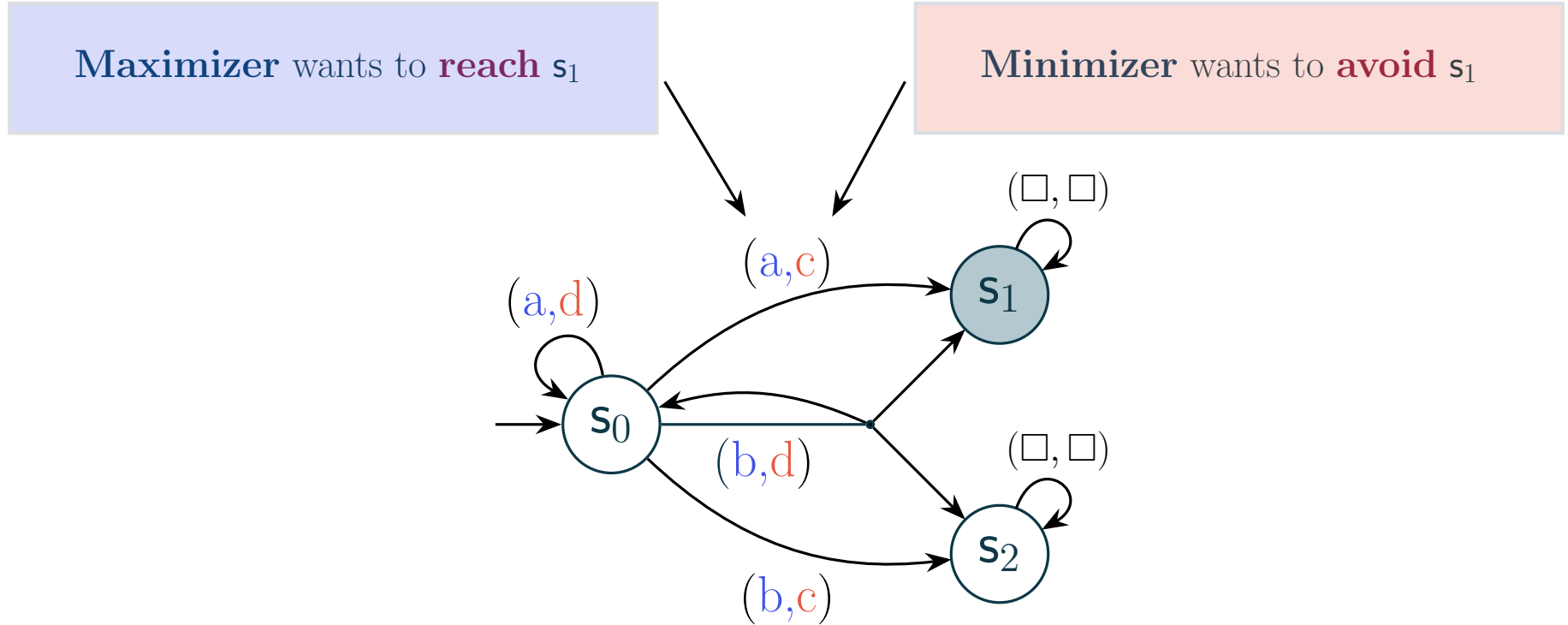


Concurrent Stochastic Reachability Games

Maximizer wants to **reach** s_1



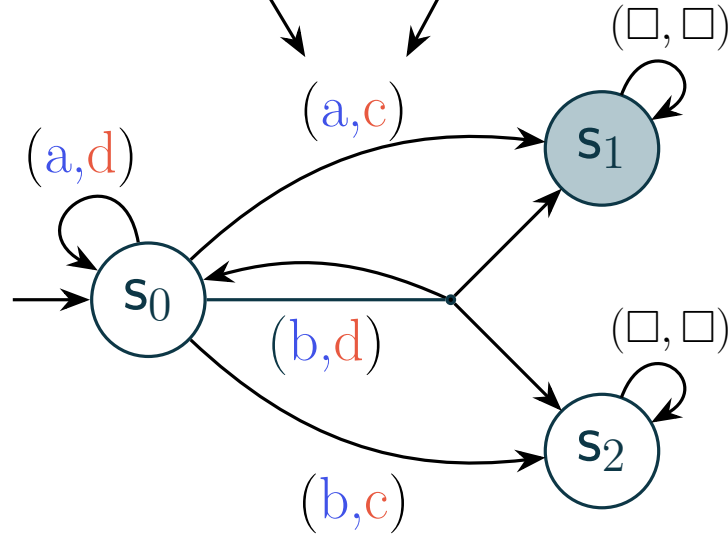
Concurrent Stochastic Reachability Games



Concurrent Stochastic Reachability Games

Maximizer wants to **reach** s_1

Minimizer wants to **avoid** s_1



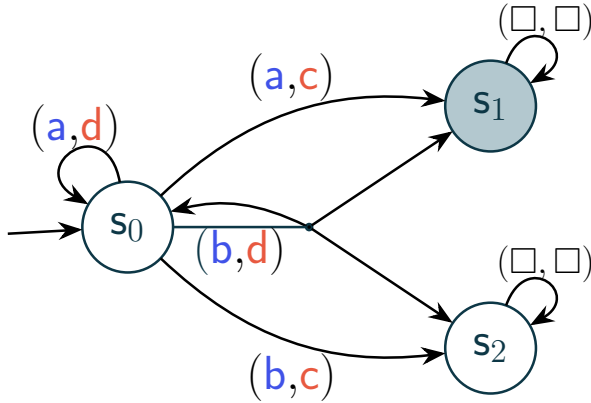
Value: Supremum probability of reaching s_1

Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)

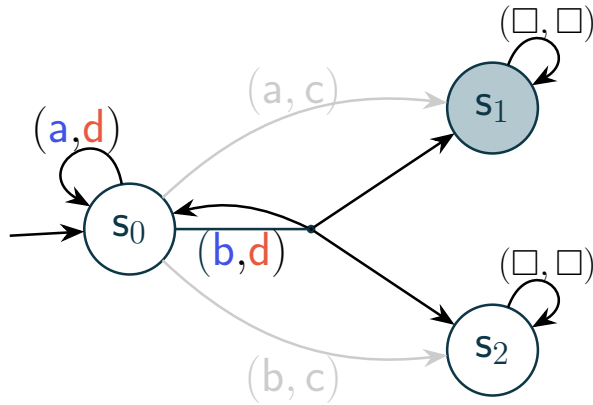
Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)



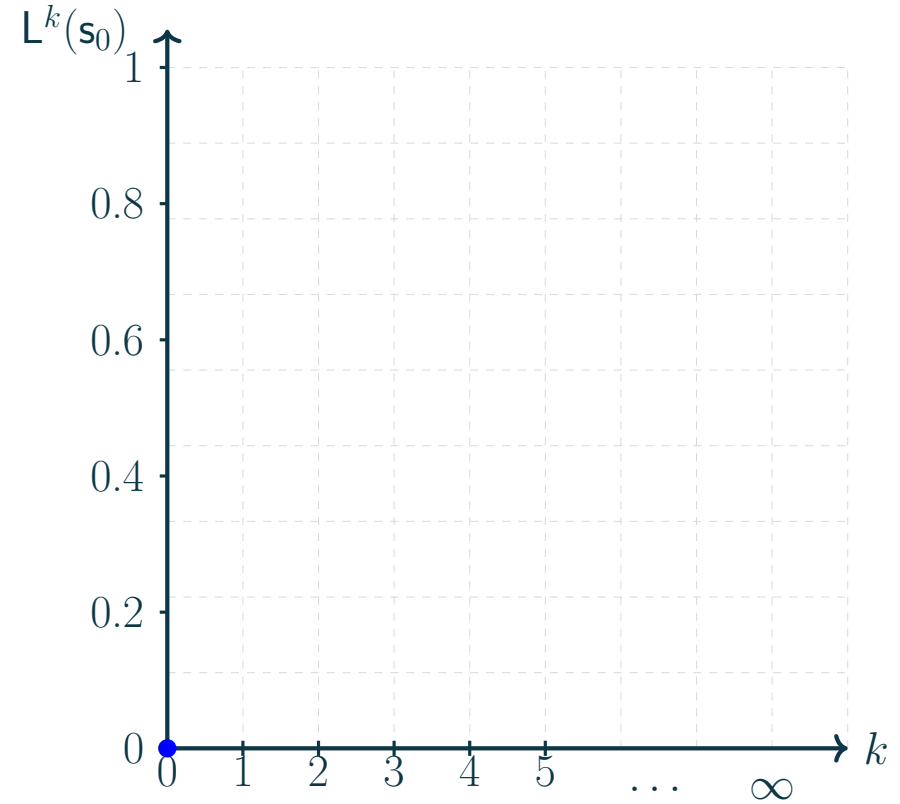
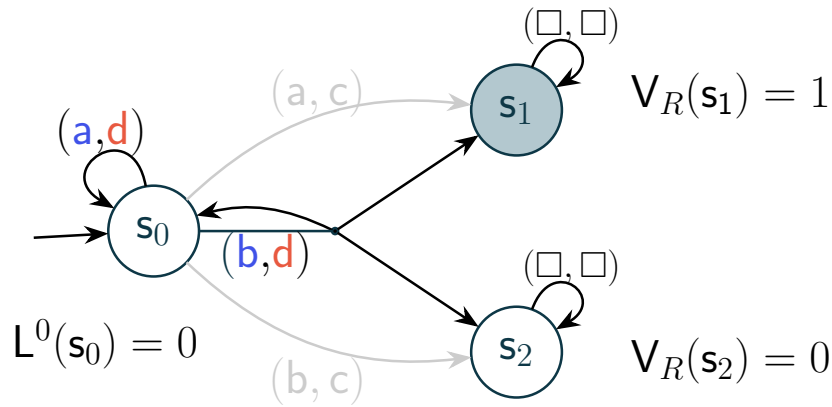
Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)



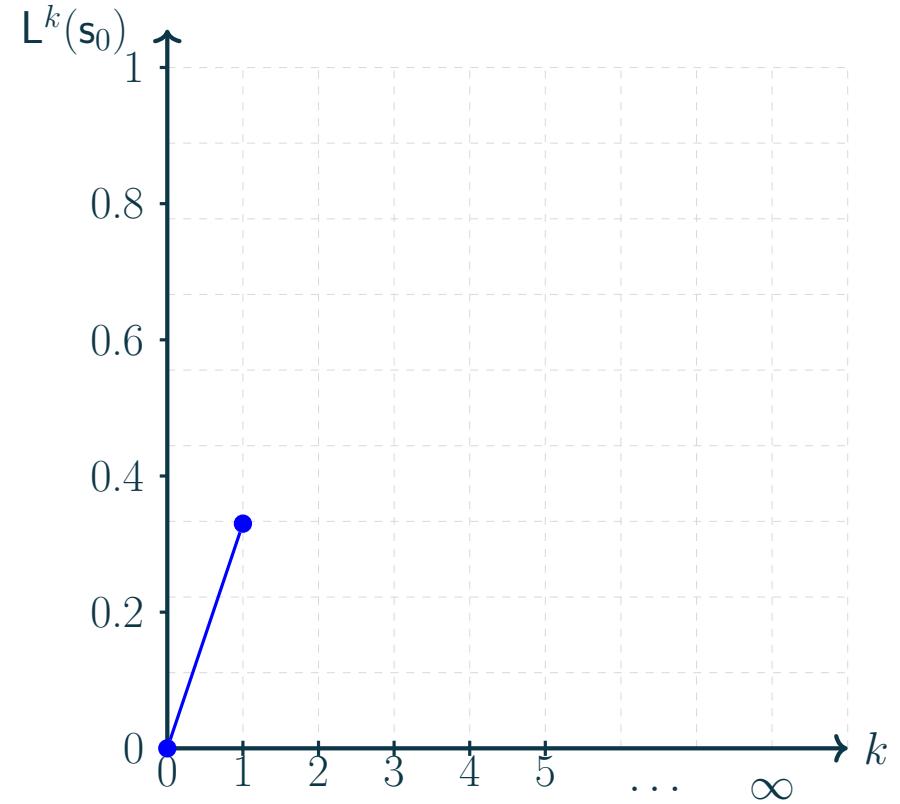
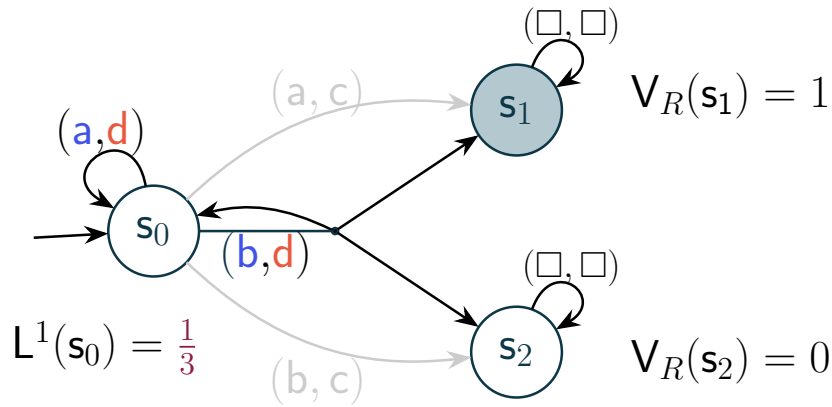
Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)



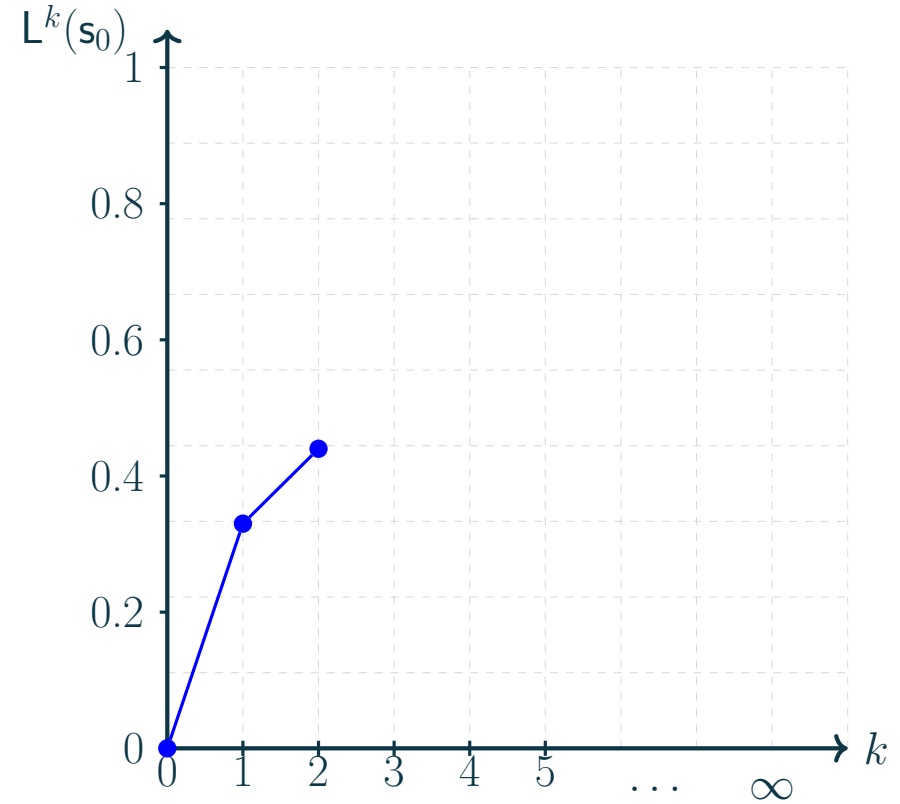
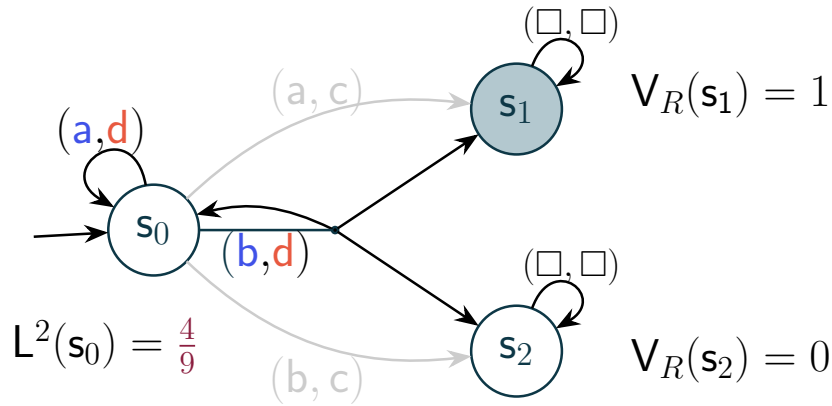
Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)



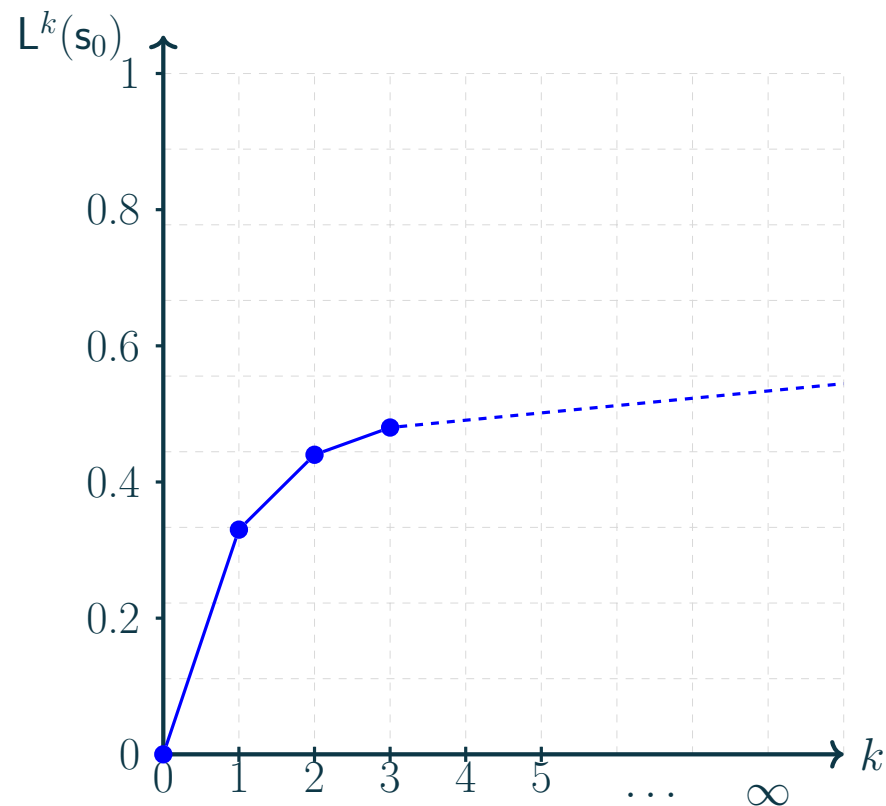
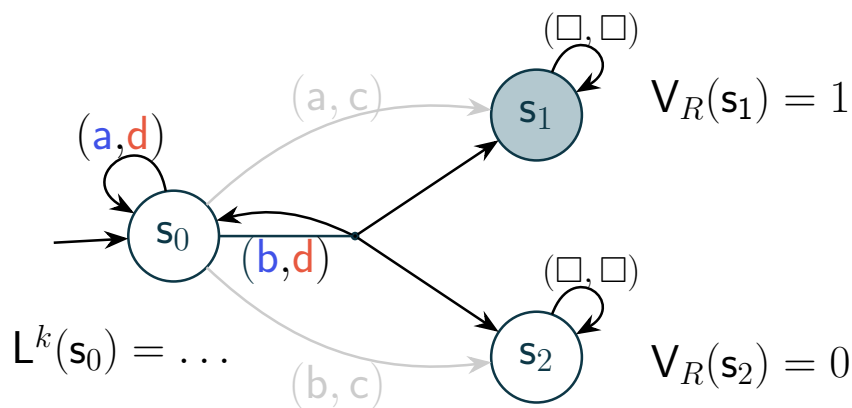
Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)



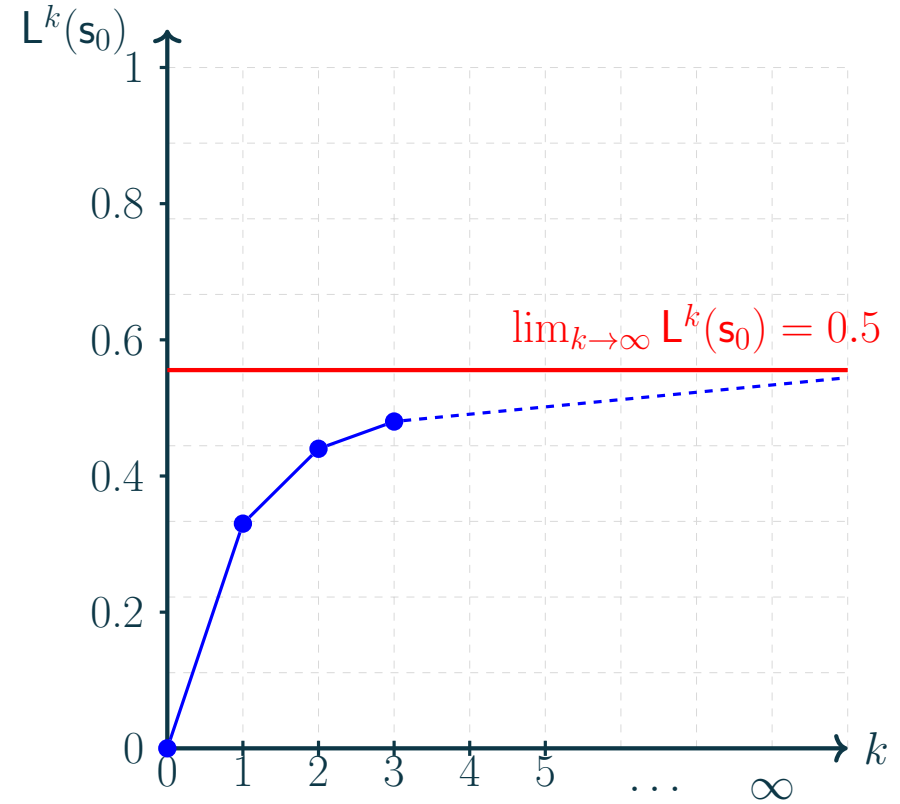
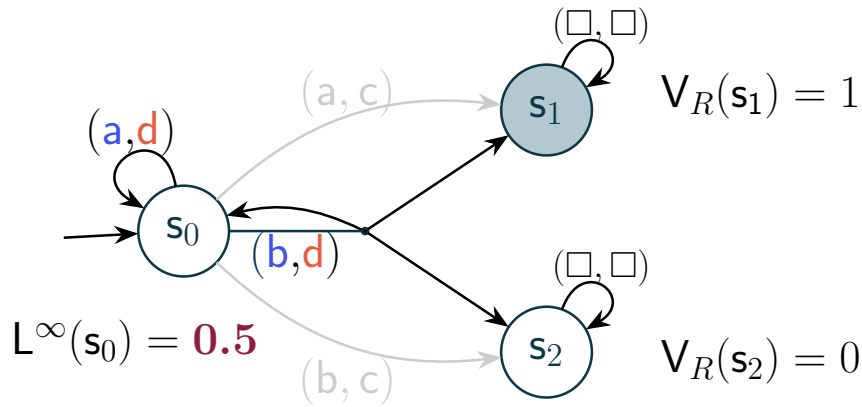
Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)



Calculating the Value

- **Initialize** with **lowest** possible value and **iteratively update** (using Bellman update $\mathfrak{B}$)

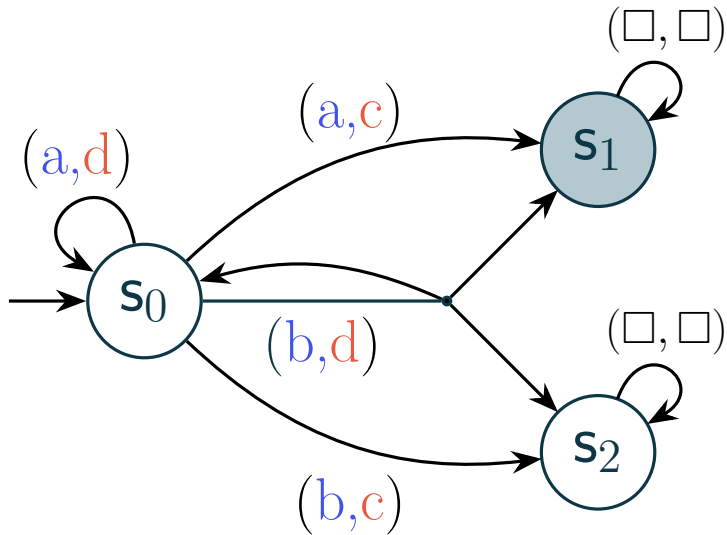


$$\text{CSG} + \text{a strategy} = \text{MDP}$$

- **Strategy:** Distribution over actions

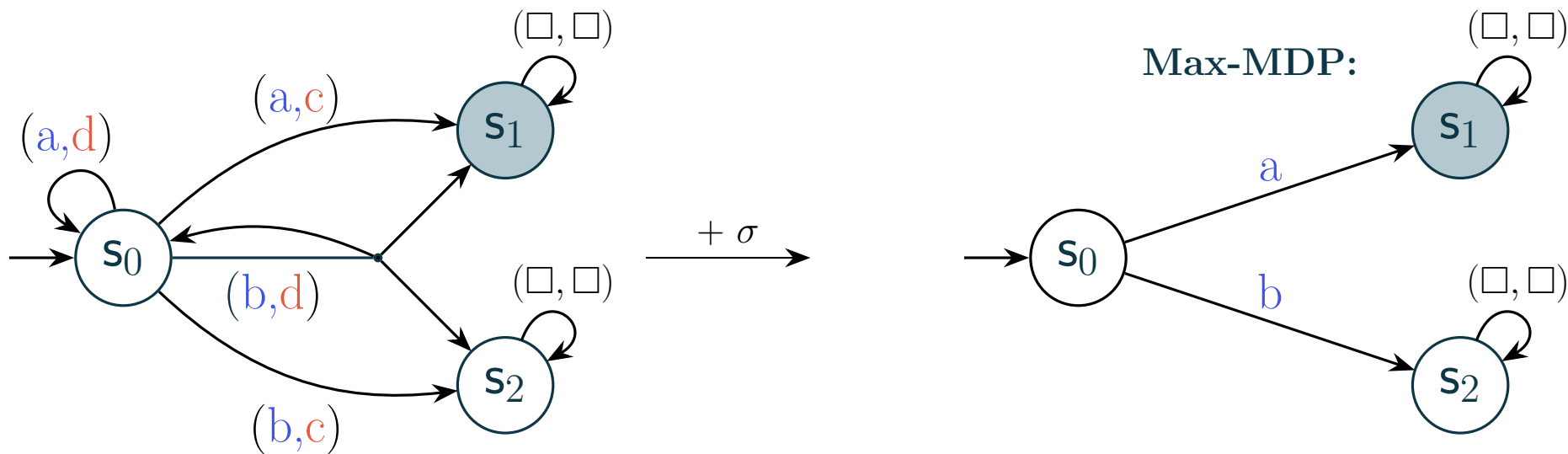
CSG + a strategy = MDP

- **Strategy:** Distribution over actions
- E.g.: $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$



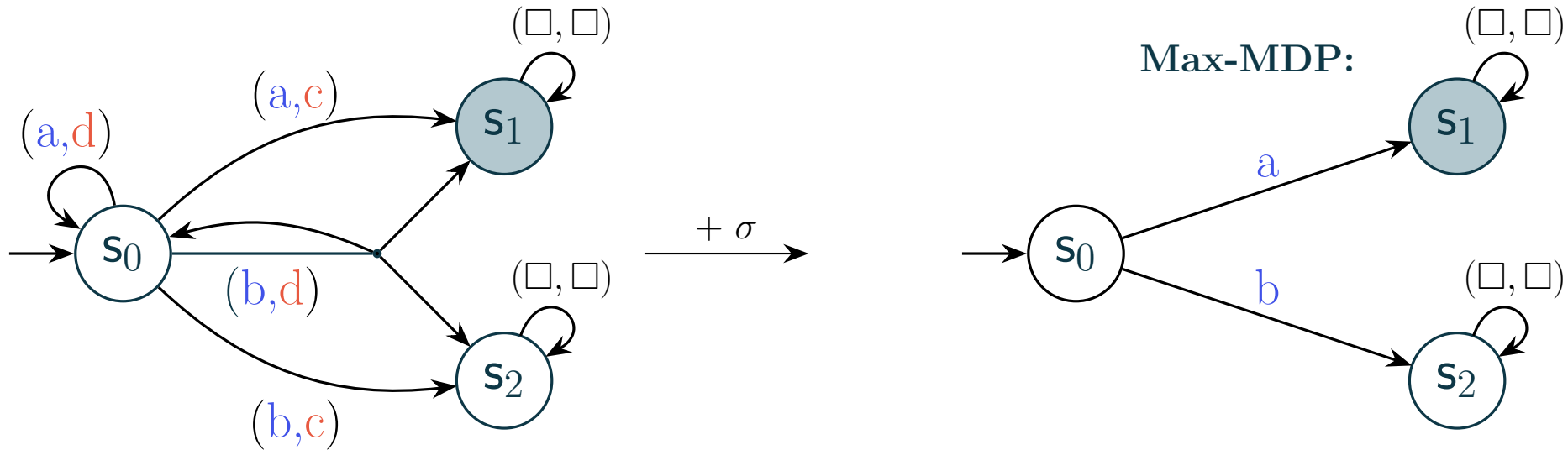
CSG + a strategy = MDP

- **Strategy:** Distribution over actions
- E.g.: $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$



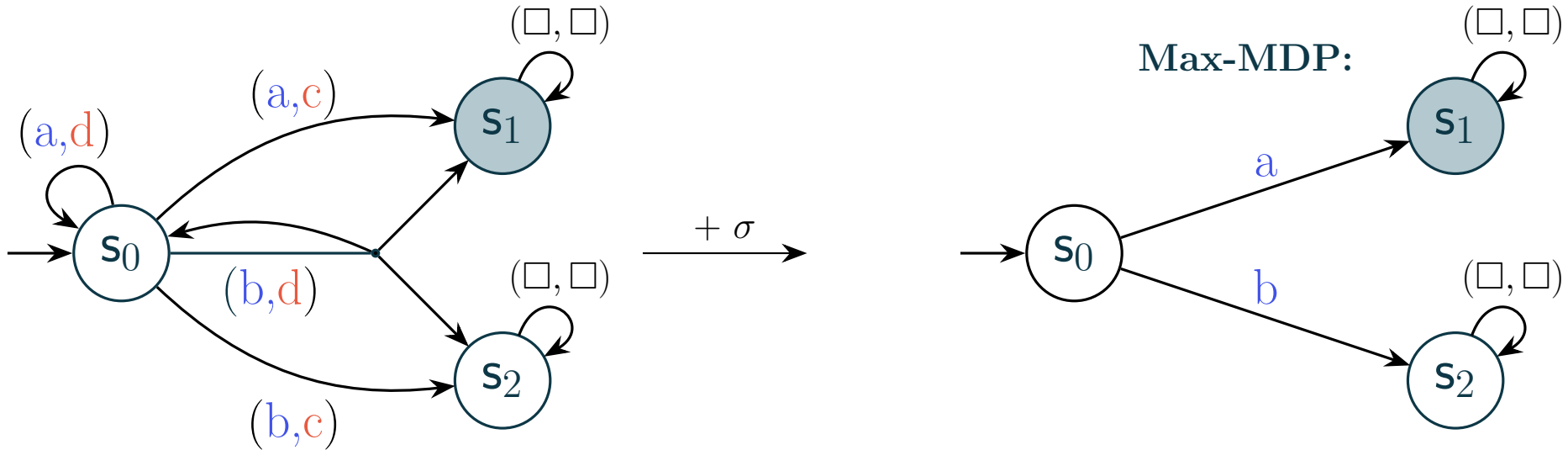
CSG + a strategy = MDP

- **Strategy:** Distribution over actions
- E.g.: $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$
- **Max-MDP:** Values **over-approximate** the values attainable in the CSG



CSG + a strategy = MDP

- **Strategy:** Distribution over actions
- E.g.: $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$
- **Max-MDP:** Values **over-approximate** the values attainable in the CSG
- **Min-MDP:** Values **under-approximate** the values attainable in the CSG



Finding Optimal Strategy using Strategy Iteration

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions
```

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions  
2  DO  
3     $\rho \leftarrow \rho'$ 
```

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions  
2  DO  
3     $\rho \leftarrow \rho'$   
4     $M^\rho \leftarrow \text{CONSTRUCT\_MDP}(\text{CSG } G, \text{strategy } \rho)$ 
```

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions
2  DO
3     $\rho \leftarrow \rho'$ 
4     $\mathbf{M}^\rho \leftarrow \text{CONSTRUCT\_MDP}(\text{CSG } \mathbf{G}, \text{strategy } \rho)$ 
5     $\mathcal{L} \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^\rho)$     // e.g. using value iteration
```

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions
2  DO
3     $\rho \leftarrow \rho'$ 
4     $\mathbf{M}^\rho \leftarrow \text{CONSTRUCT\_MDP}(\text{CSG } \mathbf{G}, \text{strategy } \rho)$ 
5     $\mathcal{L} \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^\rho)$     // e.g. using value iteration
6     $\mathbf{l} \leftarrow \{\mathbf{s} \in \mathbf{S} \mid \mathfrak{B}(\mathcal{L})(\mathbf{s}) > \mathcal{L}(\mathbf{s})\}$  // set of states where  $\rho$  can be improved
```

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions
2  DO
3     $\rho \leftarrow \rho'$ 
4     $\mathbf{M}^\rho \leftarrow \text{CONSTRUCT\_MDP}(\text{CSG } \mathbf{G}, \text{strategy } \rho)$ 
5     $\mathcal{L} \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^\rho)$     // e.g. using value iteration
6     $\mathbf{I} \leftarrow \{\mathbf{s} \in \mathbf{S} \mid \mathfrak{B}(\mathcal{L})(\mathbf{s}) > \mathcal{L}(\mathbf{s})\}$  // set of states where  $\rho$  can be improved
7     $\forall \mathbf{s} \in \mathbf{I} : \rho'(\mathbf{s}) \leftarrow \text{UPDATE\_STRATEGY}()$ 
```

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions
2  DO
3     $\rho \leftarrow \rho'$ 
4     $\mathbf{M}^\rho \leftarrow \text{CONSTRUCT\_MDP}(\text{CSG } \mathbf{G}, \text{strategy } \rho)$ 
5     $\mathcal{L} \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^\rho)$     // e.g. using value iteration
6     $\mathbf{l} \leftarrow \{\mathbf{s} \in \mathbf{S} \mid \mathfrak{B}(\mathcal{L})(\mathbf{s}) > \mathcal{L}(\mathbf{s})\}$  // set of states where  $\rho$  can be improved
7     $\forall \mathbf{s} \in \mathbf{l} : \rho'(\mathbf{s}) \leftarrow \text{UPDATE\_STRATEGY}()$ 
8  WHILE  $\rho' \neq \rho$ 
9  RETURN  $\rho'$ 
```

Finding Optimal Strategy using Strategy Iteration

```
1   $\rho' \leftarrow \text{INIT\_STRATEGY}()$            // e.g. uniform distribution over actions
2  DO
3     $\rho \leftarrow \rho'$ 
4     $\mathbf{M}^\rho \leftarrow \text{CONSTRUCT\_MDP}(\text{CSG } \mathbf{G}, \text{strategy } \rho)$ 
5     $\mathcal{L} \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^\rho)$     // e.g. using value iteration
6     $\mathbf{I} \leftarrow \{\mathbf{s} \in \mathbf{S} \mid \mathfrak{B}(\mathcal{L})(\mathbf{s}) > \mathcal{L}(\mathbf{s})\}$  // set of states where  $\rho$  can be improved
7     $\forall \mathbf{s} \in \mathbf{I} : \rho'(\mathbf{s}) \leftarrow \text{UPDATE\_STRATEGY}()$ 
8  WHILE  $\rho' \neq \rho$ 
9  RETURN  $\rho'$ 
```

Problem: In CSGs optimal strategy for Maximizer might not exist

Bounded Strategy Iteration

- Iteratively improve Maximizer's and Minimizer's strategy

Bounded Strategy Iteration

- Iteratively improve Maximizer's and Minimizer's strategy

```
...    // init strategies, construct MDPs, calculate values
DO
     $k \leftarrow k + 1$ 
     $\mathcal{L} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) > \mathcal{L}^{k-1}(s)\}$ 
```

Bounded Strategy Iteration

- Iteratively improve Maximizer's **and** Minimizer's strategy

```
...    // init strategies, construct MDPs, calculate values
DO
     $k \leftarrow k + 1$ 
     $\mathcal{L} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) > \mathcal{L}^{k-1}(s)\}$ 
     $\mathcal{U} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{U}^{k-1})(s) < \mathcal{U}^{k-1}(s)\}$ 
```

Bounded Strategy Iteration

- Iteratively improve Maximizer's **and** Minimizer's strategy

```
...    // init strategies, construct MDPs, calculate values
DO
     $k \leftarrow k + 1$ 
     $I^{\mathcal{L}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) > \mathcal{L}^{k-1}(s)\}$ 
     $I^{\mathcal{U}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{U}^{k-1})(s) < \mathcal{U}^{k-1}(s)\}$ 
     $\forall s \in I^{\mathcal{L}} : \rho^k(s) \leftarrow \text{UPDATE\_MAX\_STRATEGY}()$ 
```

Bounded Strategy Iteration

- Iteratively improve Maximizer's and Minimizer's strategy

```
...    // init strategies, construct MDPs, calculate values
DO
     $k \leftarrow k + 1$ 
     $I^{\mathcal{L}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) > \mathcal{L}^{k-1}(s)\}$ 
     $I^{\mathcal{U}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{U}^{k-1})(s) < \mathcal{U}^{k-1}(s)\}$ 
     $\forall s \in I^{\mathcal{L}} : \rho^k(s) \leftarrow \text{UPDATE\_MAX\_STRATEGY}()$ 
     $\forall s \in I^{\mathcal{U}} : \sigma^k(s) \leftarrow \text{UPDATE\_MIN\_STRATEGY}()$ 
```

Bounded Strategy Iteration

- Iteratively improve Maximizer's and Minimizer's strategy

```
...    // init strategies, construct MDPs, calculate values
DO
     $k \leftarrow k + 1$ 
     $I^{\mathcal{L}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) > \mathcal{L}^{k-1}(s)\}$ 
     $I^{\mathcal{U}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{U}^{k-1})(s) < \mathcal{U}^{k-1}(s)\}$ 
     $\forall s \in I^{\mathcal{L}} : \rho^k(s) \leftarrow \text{UPDATE\_MAX\_STRATEGY}()$ 
     $\forall s \in I^{\mathcal{U}} : \sigma^k(s) \leftarrow \text{UPDATE\_MIN\_STRATEGY}()$ 
     $\mathcal{U}^k, \mathcal{L}^k \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^{\sigma^k}, \mathbf{M}^{\rho^k})$ 
```

Bounded Strategy Iteration

- Iteratively improve Maximizer's and Minimizer's strategy

```
...    // init strategies, construct MDPs, calculate values
DO
     $k \leftarrow k + 1$ 
     $I^{\mathcal{L}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) > \mathcal{L}^{k-1}(s)\}$ 
     $I^{\mathcal{U}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{U}^{k-1})(s) < \mathcal{U}^{k-1}(s)\}$ 
     $\forall s \in I^{\mathcal{L}} : \rho^k(s) \leftarrow \text{UPDATE\_MAX\_STRATEGY}()$ 
     $\forall s \in I^{\mathcal{U}} : \sigma^k(s) \leftarrow \text{UPDATE\_MIN\_STRATEGY}()$ 
     $\mathcal{U}^k, \mathcal{L}^k \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^{\sigma^k}, \mathbf{M}^{\rho^k})$ 
WHILE  $(\mathcal{U}^k - \mathcal{L}^k > \varepsilon)$ 
```

Bounded Strategy Iteration

- Iteratively improve Maximizer's and Minimizer's strategy

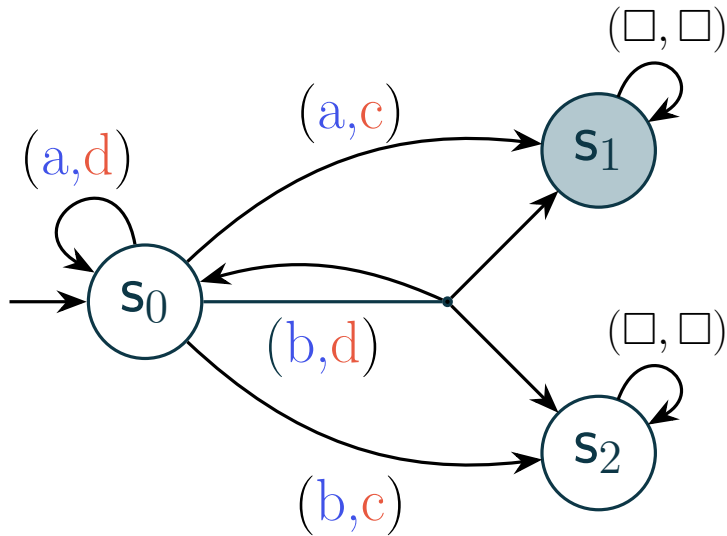
```
...    // init strategies, construct MDPs, calculate values
DO
   $k \leftarrow k + 1$ 
   $I^{\mathcal{L}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) > \mathcal{L}^{k-1}(s)\}$ 
   $I^{\mathcal{U}} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{U}^{k-1})(s) < \mathcal{U}^{k-1}(s)\}$ 
   $\forall s \in I^{\mathcal{L}} : \rho^k(s) \leftarrow \text{UPDATE\_MAX\_STRATEGY}()$ 
   $\forall s \in I^{\mathcal{U}} : \sigma^k(s) \leftarrow \text{UPDATE\_MIN\_STRATEGY}()$ 
   $\mathcal{U}^k, \mathcal{L}^k \leftarrow \text{COMPUTE\_VALUES}(\mathbf{M}^{\sigma^k}, \mathbf{M}^{\rho^k})$ 
WHILE ( $\mathcal{U}^k - \mathcal{L}^k > \varepsilon$ )
```

Problem: End components

End Components Problem

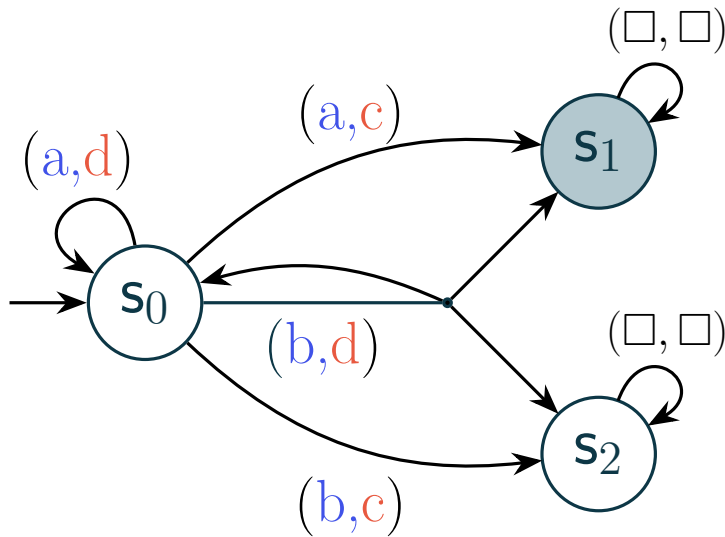
End Components Problem

End Component: Set of states where staying forever is possible



End Components Problem

End Component: Set of states where staying forever is possible

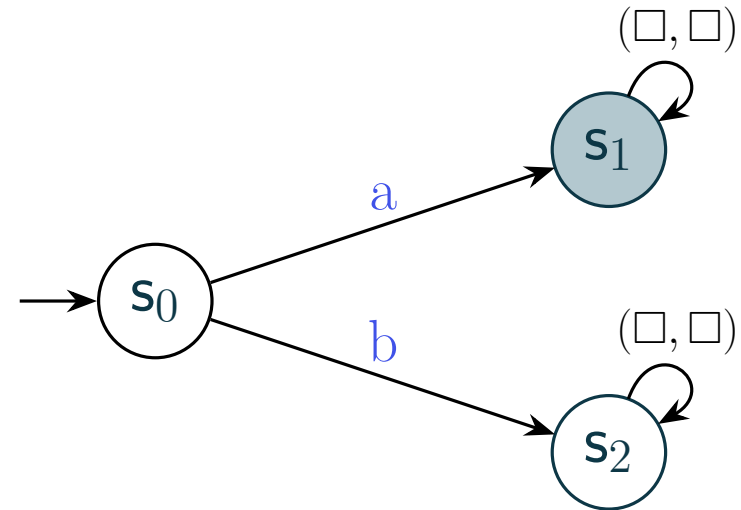
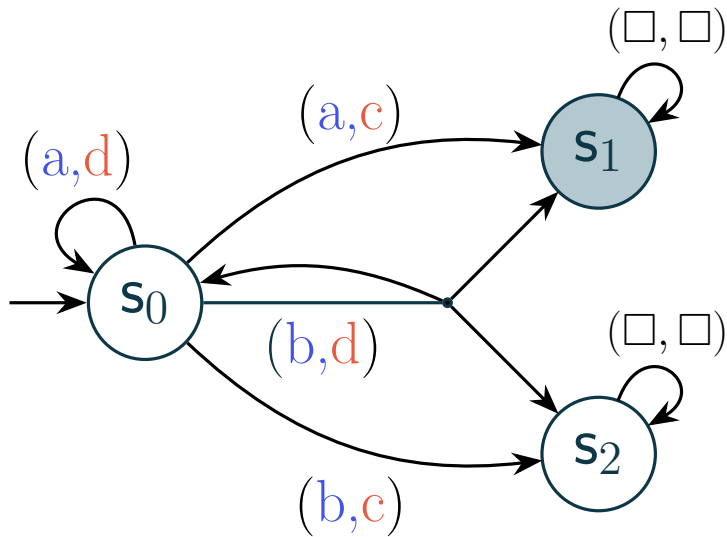


- Recall fixing $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$

End Components Problem

End Component: Set of states where staying forever is possible

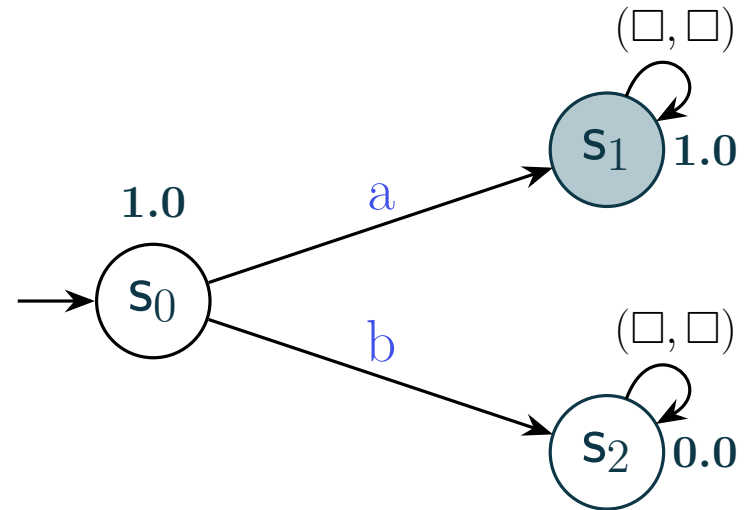
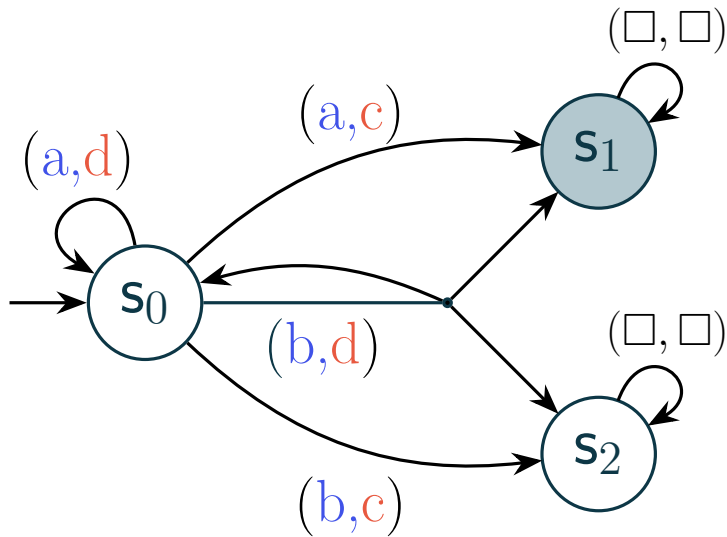
- Recall fixing $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$



End Components Problem

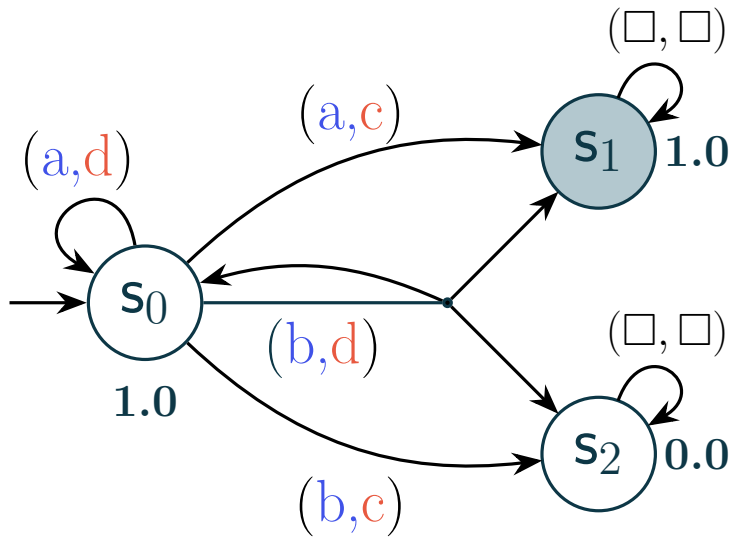
End Component: Set of states where staying forever is possible

- Recall fixing $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$

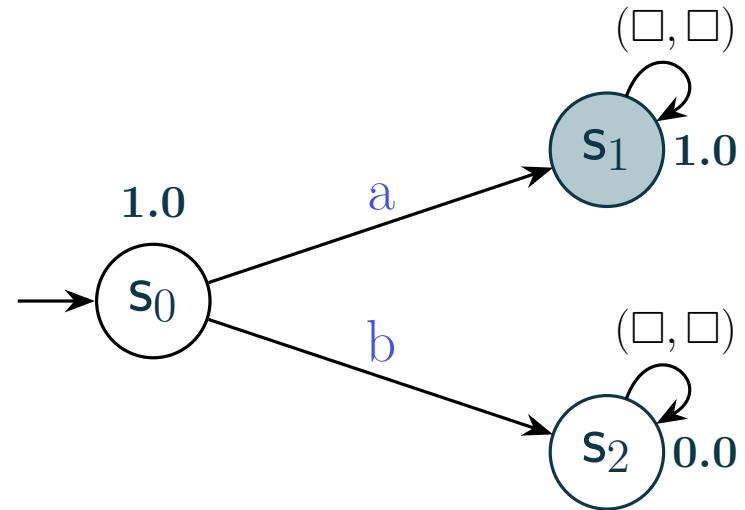


End Components Problem

End Component: Set of states where staying forever is possible

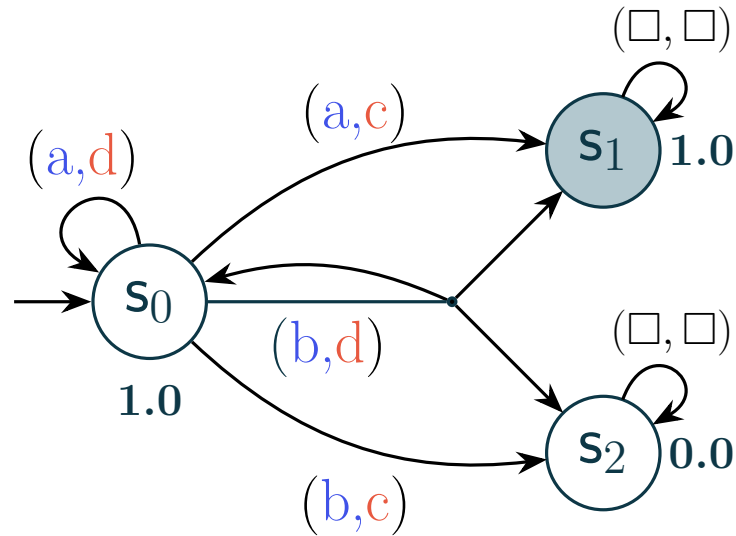


- Recall fixing $\sigma := \{c \mapsto 1.0, d \mapsto 0.0\}$



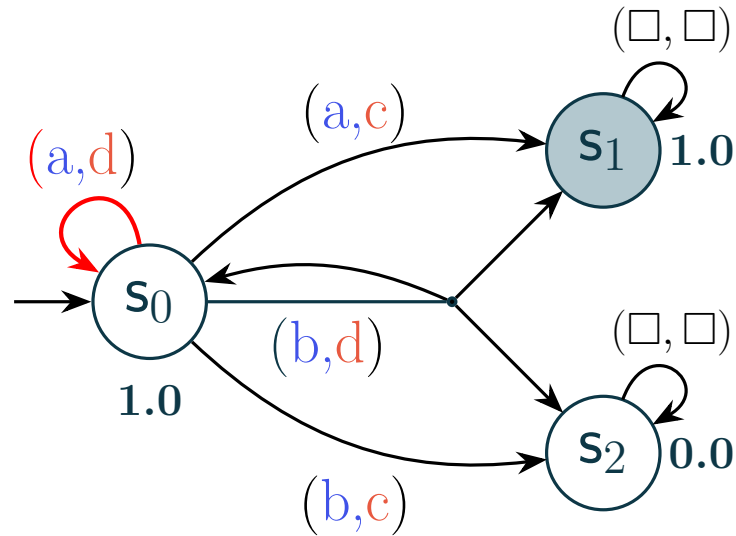
End Components Problem

End Component: Set of states where staying forever is possible



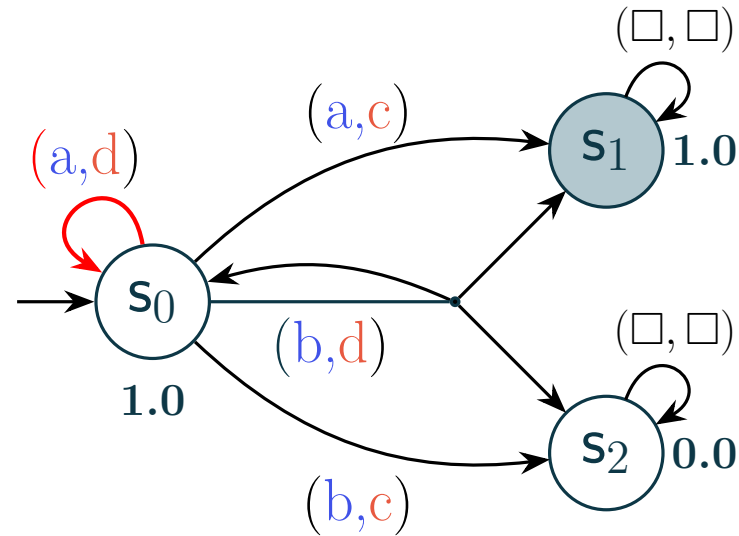
End Components Problem

End Component: Set of states where staying forever is possible



End Components Problem

End Component: Set of states where staying forever is possible



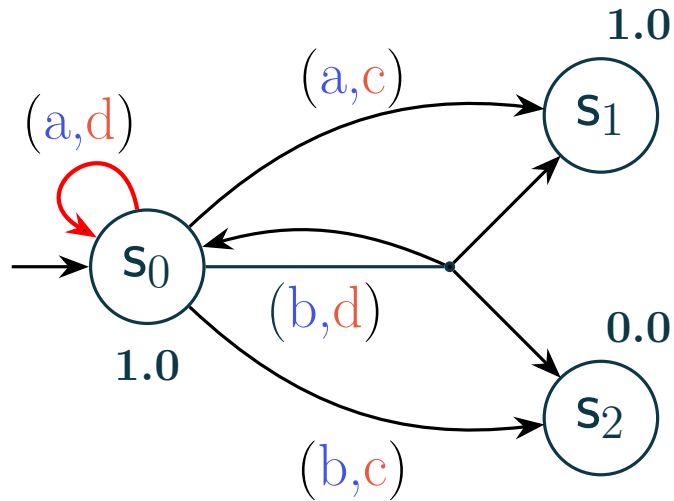
$\mathcal{U} = \emptyset \Rightarrow$ No strategy update $\Rightarrow$ **we are stuck!**

Idea: Use Deflation

- Deflating solves the end components problem (for CSGs [LICS'25])

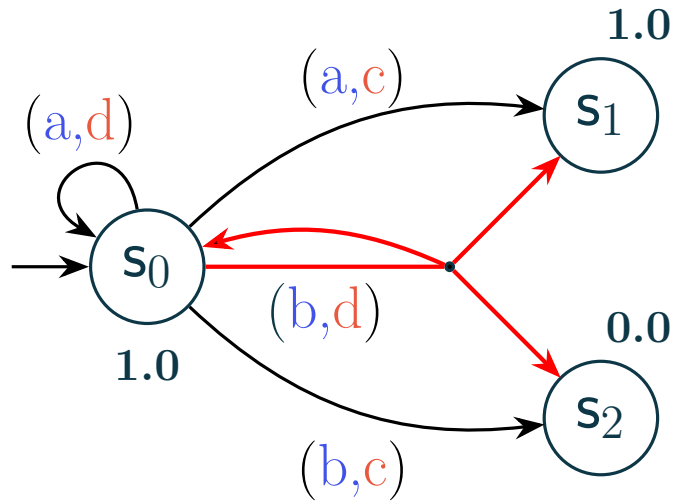
Idea: Use Deflation

- Deflating solves the end components problem (for CSGs [LICS'25])



Idea: Use Deflation

- Deflating solves the end components problem (for CSGs [LICS'25])



Idea: Use Deflation

- Deflating solves the end components problem (for CSGs [LICS'25])

...

DO

$$k \leftarrow k + 1$$

$$|\mathcal{L} \leftarrow \{s \in S \mid \mathfrak{B}(\mathcal{L}^{k-1})(s) \geq \mathcal{L}^{k-1}(s)\}$$

$$|\mathcal{U} \leftarrow \{s \in S \mid (\mathfrak{D} \circ \mathfrak{B})(\mathcal{U}^{k-1})(s) \leq \mathcal{U}^{k-1}(s)\}$$

$$\forall s \in |\mathcal{L} : \rho^k(s) \leftarrow \text{UPDATE_MAX_STRATEGY}()$$

$$\forall s \in |\mathcal{U} : \sigma^k(s) \leftarrow \text{UPDATE_MIN_STRATEGY}()$$

$$\mathcal{U}^k, \mathcal{L}^k \leftarrow \text{COMPUTE_VALUES}(\mathbf{M}^{\sigma^k}, \mathbf{M}^{\rho^k})$$

WHILE $(\mathcal{U}^k - \mathcal{L}^k > \varepsilon)$

Correctness

Correctness

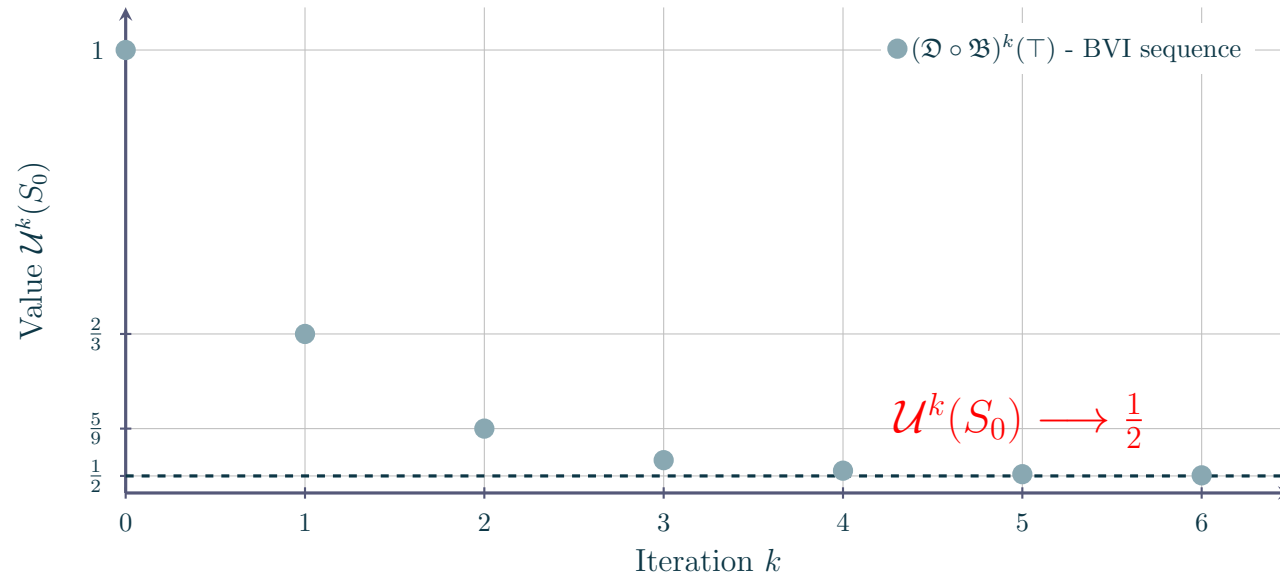
- All $\mathcal{U}^k$ are greater than the true value

Correctness

- All $\mathcal{U}^k$ are greater than the true value
- The sequence of valuations $(\mathcal{U}^k)_{k \in \mathbb{N}}$ is monotonically decreasing, and

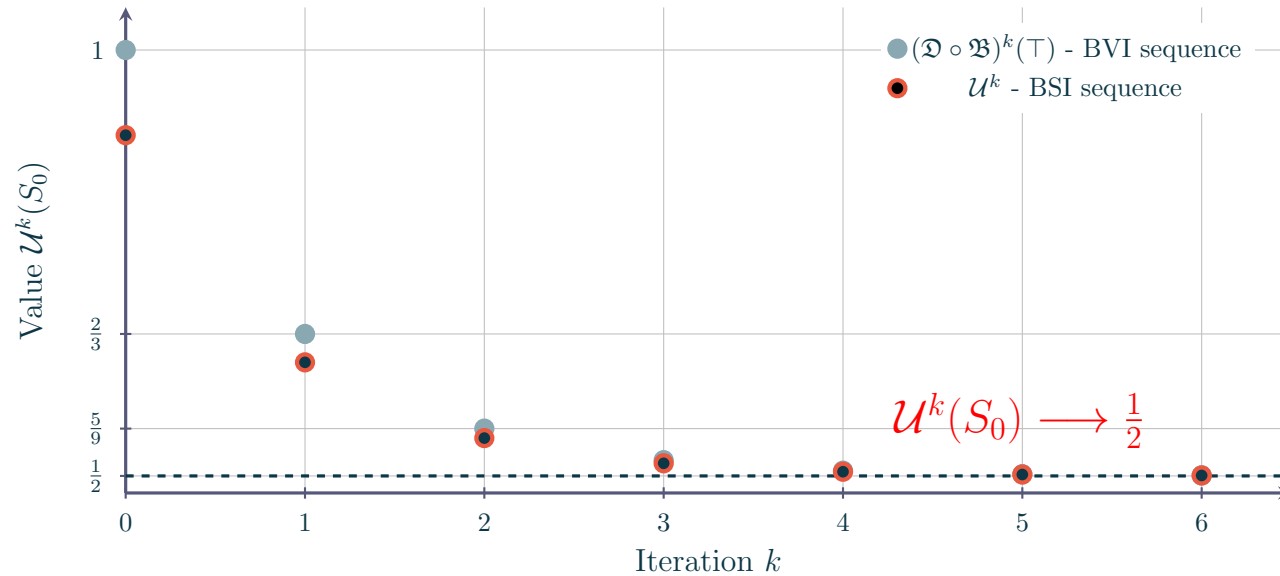
Correctness

- All $\mathcal{U}^k$ are greater than the true value
- The sequence of valuations $(\mathcal{U}^k)_{k \in \mathbb{N}}$ is monotonically decreasing, and



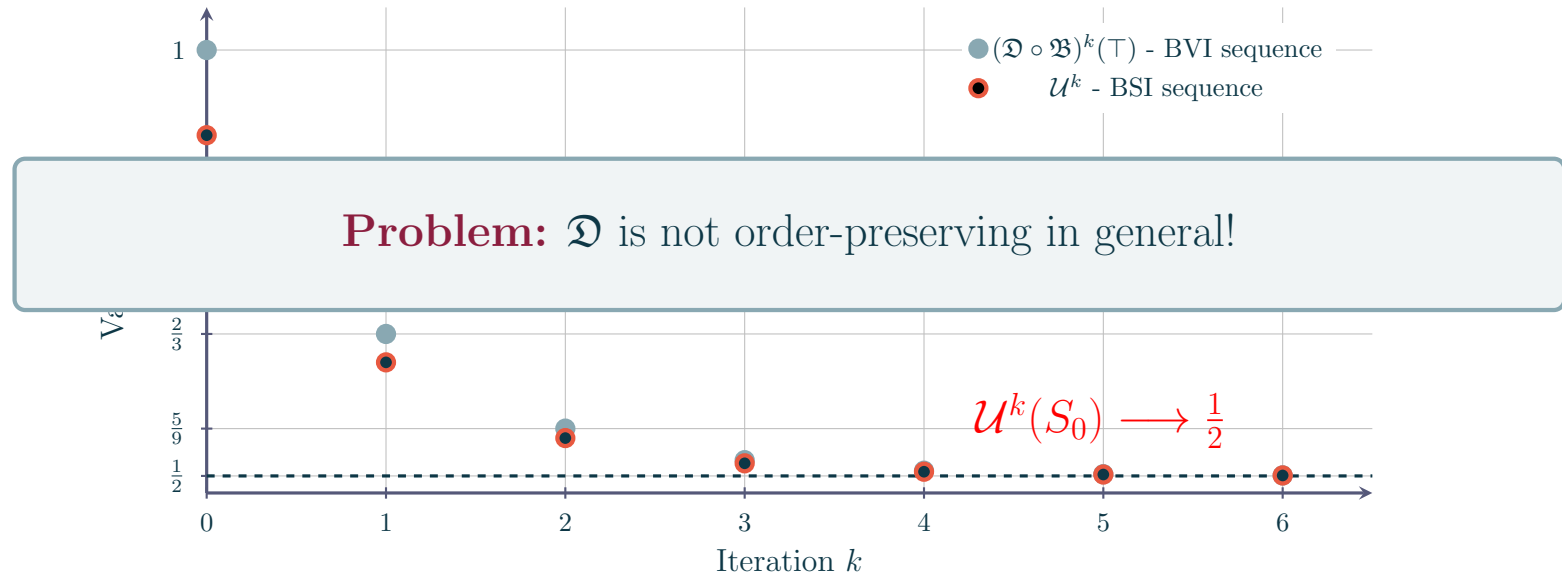
Correctness

- All $\mathcal{U}^k$ are greater than the true value
- The sequence of valuations $(\mathcal{U}^k)_{k \in \mathbb{N}}$ is monotonically decreasing, and



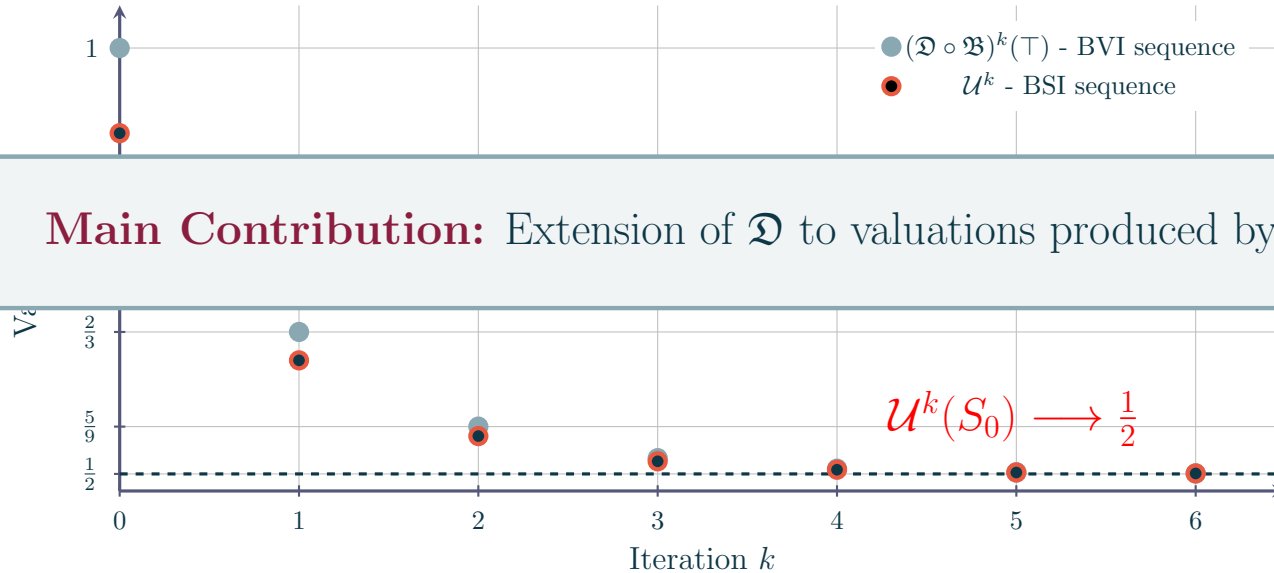
Correctness

- All $\mathcal{U}^k$ are greater than the true value
- The sequence of valuations $(\mathcal{U}^k)_{k \in \mathbb{N}}$ is monotonically decreasing, and



Correctness

- All $\mathcal{U}^k$ are greater than the true value
- The sequence of valuations $(\mathcal{U}^k)_{k \in \mathbb{N}}$ is monotonically decreasing, and



Summary

Summary

- Goal: a **bounded strategy iteration** algorithm for CSGs

Summary

- Goal: a **bounded strategy iteration** algorithm for CSGs
- BSI **does not work out of the box**: end components obstruct the upper bound

Summary

- Goal: a **bounded strategy iteration** algorithm for CSGs
- BSI **does not work out of the box**: end components obstruct the upper bound
- Natural idea: combine strategy iteration with **deflation** from bounded value iteration

Summary

- Goal: a **bounded strategy iteration** algorithm for CSGs
- BSI **does not work out of the box**: end components obstruct the upper bound
- Natural idea: combine strategy iteration with **deflation** from bounded value iteration
- However, the existing deflation is order-preserving only on BVI-generated valuations

Summary

- Goal: a **bounded strategy iteration** algorithm for CSGs
- BSI **does not work out of the box**: end components obstruct the upper bound
- Natural idea: combine strategy iteration with **deflation** from bounded value iteration
- However, the existing deflation is order-preserving only on BVI-generated valuations

Main contribution

Extend the theory of deflation to the valuations produced by BSI

Summary

- Goal: a **bounded strategy iteration** algorithm for CSGs
- BSI **does not work out of the box**: end components obstruct the upper bound
- Natural idea: combine strategy iteration with **deflation** from bounded value iteration
- However, the existing deflation is order-preserving only on BVI-generated valuations

Main contribution

Extend the theory of deflation to the valuations produced by BSI

Thank You!